Title: Visual Analysis of Dynamics Behaviour of an Iterative Method Depending on Selected Parameters and Modifications

Author: Ireneusz Gościniak, Krzysztof Gdawiec

Citation style: Gościniak Ireneusz, Gdawiec Krzysztof. (2020). Visual Analysis of Dynamics Behaviour of an Iterative Method Depending on Selected Parameters and Modifications. "Entropy" Vol. 22, iss. 7 (2020), art. no 734 (early access), doi 10.3390/e22070734

entropy

MDPI

Article

Visual Analysis of Dynamics Behaviour of an Iterative Method Depending on Selected Parameters and Modifications

Ireneusz Gościniak * and Krzysztof Gdawiec

Institute of Computer Science, University of Silesia, Będzińska 39, 41–200 Sosnowiec, Poland; krzysztof.gdawiec@us.edu.pl

* Correspondence: ireneusz.gosciniak@us.edu.pl

Received: 5 May 2020; Accepted: 29 June 2020; Published: 2 July 2020

Abstract: There is a huge group of algorithms described in the literature that iteratively find solutions of a given equation. Most of them require tuning. The article presents root-finding algorithms that are based on the Newton–Raphson method which iteratively finds the solutions, and require tuning. The modification of the algorithm implements the best position of particle similarly to the particle swarm optimisation algorithms. The proposed approach allows visualising the impact of the algorithm's elements on the complex behaviour of the algorithm. Moreover, instead of the standard Picard iteration, various feedback iteration processes are used in this research. Presented examples and the conducted discussion on the algorithm's operation allow to understand the influence of the proposed modifications on the algorithm's behaviour. Understanding the impact of the proposed modification on the algorithm's operation can be helpful in using it in other algorithms. The obtained images also have potential artistic applications.

Keywords: root finding; dynamics; iterations; visualisation

1. Introduction

Most of the algorithms determining the local extremes and especially the minimum of the function $\mathbf{F}$ are based on determining the value of the function or its gradient. In the case of such an algorithm, the new position of the particle is calculated basing on its previous position and the gradient which determine the direction of the movement [1]—an example of this approach is the gradient descent method:

$$\mathbf{z}_i' = \mathbf{z}_i - \gamma \nabla \mathbf{F}(\mathbf{z}_i), \tag{1}$$

where: $-\nabla\mathbf{F}$—negative gradient, γ—step size, $\mathbf{z}'_i$—the current position of the ith particle in a D dimensional environment, $\mathbf{z}_i$—the previous position of the ith particle. There are many algorithms implementing modifications of this method, and its operation is well described in the literature.

Evolutionary algorithms are another group of algorithms used to solve optimisation problems, including minimisation. As shown in [2], the analysis of particle behaviour in the evolutionary algorithms is a very complex problem. In practice, many algorithms similar to the evolutionary algorithms are used [3]. Particularly noteworthy is the group of particle swarm optimisation algorithms (PSO) [4]. The complex nature of the interaction between particles does not allow us to determine their impact on the operation of the algorithm in a simple way. The behaviour of particles in the PSO algorithm can be described by the following equation:

$$\mathbf{z}_i' = \mathbf{z}_i + \mathbf{v}_i', \tag{2}$$

where: $\mathbf{z}'_i$—the current position of the ith particle in a D dimensional environment, $\mathbf{z}_i$—the previous position of the ith particle, $\mathbf{v}'_i$—the current velocity of the ith particle in a D dimensional environment that is given by the following formula:

$$\mathbf{v}'_i = \omega \mathbf{v}_i + \eta_1 r_1 (\mathbf{z}_{pbest\ i} - \mathbf{z}_i) + \eta_2 r_2 (\mathbf{z}_{gbest} - \mathbf{z}_i), \tag{3}$$

where: $\mathbf{v}_i$—the previous velocity of the ith particle, ω—inertia weight ($\omega \in [0,1]$), η_1, η_2—acceleration constants ($\eta_1, \eta_2 \in (0,1]$), r_1, r_2—random numbers generated uniformly in the $[0,1]$ interval, $\mathbf{z}_{pbest\ i}$—the best position of the ith particle, $\mathbf{z}_{gbest}$—the global best position of the particles. The best position of the particle and the best global position are calculated in each iteration. The position of the particle and the best global position have a significant impact on the behaviour of not only the particle, but the entire swarm, directing them towards the best solution. The particle behaviour is determined by three parameters: inertia weight (ω) and two acceleration constants (η_1, η_2). The PSO algorithm belongs to the group of stochastic algorithms for which the convergence can be proved [5]. The ω parameter informs about the influence of the particle velocity at the moment $t-1$ on its velocity at the moment t. On the other hand, the η parameter affects the changes in the particle acceleration at the time t. For $\omega < 1$ the particles slow down to reach the velocity of zero. For the value of ω close to 0.7 and the value of η_1 and η_2 close to 1.5 the PSO algorithm has good convergence. The cooperation of particles releases an adaptive mechanism that drives the particle and determines its dynamics. The selection of the proper values of inertia weight and acceleration constants is a very important problem for the algorithm [6]—the algorithm tuning is intuitive and based on developer's experience.

Another important problem that arises in practice is finding the solutions of a system of non-linear equations. Due to many difficulties in solving such systems by using analytical methods many numerical methods were developed. The most famous method of this type is the Newton's method [7]. In recent years methods with different approaches were proposed. In [8] Goh et al. inspired by the simplex algorithm, proposed a partial Newton's method. Despite this method is based on Newton's method, it has slower convergence. Another method based on Newton's method was proposed in [9]. In this method Saheya et al. revised the Jacobian matrix by a rank one matrix each iteration. Sharma et al. in [10] introduced a three-step iterative scheme that is based on Newton–Traub's method. A method that is based on a Jarratt-like method was proposed by Abbasbandy et al. in [11]. This method uses only one LU factorisation which preserves and reduces computational complexities. An acceleration technique that can be used in many iterative methods was proposed in [12]. Moreover, the authors introduced a new family of two-step third order methods for solving systems of non-linear equations. Recently, Alqahtani et el. proposed a three-step family of iterative methods [13]. This family is a generalisation of the fourth-order King's family to the multidimensional case. Awwal et al. in [14] proposed a two-step iterative algorithm that is based on projection technique and solves system of monotone non-linear equations with convex constraints.

In addition to the methods based on classical root finding methods (Newton, Traub, Jarratt, etc.), in the literature we can find approaches based on the PSO algorithm and other metaheuristic algorithms. In [15] Wang et al. proposed a modification of the PSO algorithm in which the search dynamics is controlled by a traditional controller such as the PID (Proportional–Integral–Derivative) controller. A combination of Nelder–Mead simplex method and PSO algorithm was proposed in [16]. To overcome problems of being trapped in local minima and slow convergence, Jaberipour et al. proposed a new way of updating particle's position [17]. Another modification of the PSO algorithm for solving systems of non-linear equations was proposed by Li et al. in [18]. The authors proposed: (1) a new way of inertia weight selection, (2) dynamics conditions of stopping iterating, (3) calculation of the standardised number of restarting times based on reliability theory. Ibrahim and Tawhid in [19] introduced a hybridisation of cuckoo search and PSO for solving system of non-linear equations, and in [20] a hybridisation of differential evolution and the monarch butterfly optimisation. Recently, Liao et al. in [21] introduced a decomposition re-initialisation-based differential evolution to locate multiple roots of non-linear equations system. A study on different metaheuristic methods, namely

PSO, firefly algorithm and cuckoo search algorithm, in the solving of system of non-linear equation task was conducted in [22]. The study showed that the PSO algorithm obtains better results than the other two algorithms.

Dynamic analysis is a separate field of research using numerous investigation methods [23]. Among them, there is a graphical visualisation of dynamics [24]. In the analysis of the complex polynomial root finding process the polynomiography is a very popular method of visualising the dynamics [25–27]. In this method the number of iterations required to obtain the solution is visualised. The generated images show the dynamics of the root-finding process using some colour maps. Today, dynamics study of root-finding methods with polynomiography is an essential part of modern analysis of the quality of these methods [28]. It turns out that this form of dynamics study is a powerful tool in selecting optimal parameters that appear in the iterative methods for solving non-linear equations and to compare these methods [29,30]. The visualisation methods proposed in the work will also be called polynomiography because of their similarity to the primary polynomiography, despite the fact that we use any functions, not just polynomials.

This article proposes modifications of a root finding algorithm, that is based on the Newton's method, by using the inertia weight and the acceleration constant as well as the best position of the particle and the implementation of various iteration methods. The conducted research serves to visualise the impact of the proposed algorithm modifications on the behaviour of the particle. They allow conducting not only the discussion on the behaviour of a particle, but also to present images of aesthetic character and artistic meaning [31,32].

Nammanee et al. [33] introduced a strong convergence theorem for the modified Noor iterations in the framework of uniformly smooth Banach spaces. Their results allow to suppose that the proposed modifications using the best particle position will be an effective way to increase the effectiveness of algorithm's operation.

The rest of the paper is organised as follows. Section 2 introduces root finding algorithm that is based on the Newton–Raphson method. This method is used to illustrate the tuning influence on its behaviour. Next, Section 3 introduces iteration processes known in literature. Moreover, basing on these iterations we propose iterations that use the best position of the particle. Section 4 presents algorithms for creating polynomiographs. Then, in Section 5 a discussion on the research results illustrated by the obtained polynomiographs is made. Finally, Section 6 gives short concluding remarks.

2. The Algorithm

Numerous methods can be used to solve a system of D non-linear equations with D variables. One of the best known and intensively studied methods is the Newton–Raphson method [7]. Let $f_1, f_2, \ldots, f_D : \mathbb{R}^D \to \mathbb{R}$ and let

$$\mathbf{F}(z^1, z^2, \ldots, z^D) = \begin{bmatrix} f_1(z^1, z^2, \ldots, z^D) \\ f_2(z^1, z^2, \ldots, z^D) \\ \vdots \\ f_D(z^1, z^2, \ldots, z^D) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \mathbf{0}. \tag{4}$$

Let $\mathbf{F} : \mathbb{R}^D \to \mathbb{R}^D$ be a continuous function which has continuous first partial derivatives. Now, to solve the equation $\mathbf{F}(\mathbf{z}) = \mathbf{0}$, where $\mathbf{z} = [z^1, z^2, \ldots, z^D]$, by using the Newton–Raphson method, a starting point is selected $\mathbf{z}_0 = [z_0^1, z_0^2, \ldots, z_0^D]$ and then the iterative formula is used as follows:

$$\mathbf{z}_{n+1} = \mathbf{z}_n - \mathbf{J}^{-1}(\mathbf{z}_n)\mathbf{F}(\mathbf{z_n}) \quad n = 0, 1, 2, \ldots, \tag{5}$$

where:

$$\mathbf{J}(\mathbf{z}) = \begin{bmatrix} \frac{\partial f_1}{\partial z_1}(\mathbf{z}) & \frac{\partial f_1}{\partial z_2}(\mathbf{z}) & \dots & \frac{\partial f_1}{\partial z_D}(\mathbf{z}) \\ \frac{\partial f_2}{\partial z_1}(\mathbf{z}) & \frac{\partial f_2}{\partial z_2}(\mathbf{z}) & \dots & \frac{\partial f_2}{\partial z_D}(\mathbf{z}) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_D}{\partial z_1}(\mathbf{z}) & \frac{\partial f_D}{\partial z_2}(\mathbf{z}) & \dots & \frac{\partial f_D}{\partial z_D}(\mathbf{z}) \end{bmatrix} \tag{6}$$

is the Jacobian matrix of $\mathbf{F}$ and $\mathbf{J}^{-1}$ is its inverse.

By taking $\mathbf{N}(\mathbf{z}) = -\mathbf{J}^{-1}(\mathbf{z})\mathbf{F}(\mathbf{z})$ the Newton–Raphson method can be described as follows:

$$\mathbf{z}_{n+1} = \mathbf{z}_n + \mathbf{N}(\mathbf{z}_n), \quad n = 0, 1, 2, \dots. \tag{7}$$

To solve (4), the following algorithm can be used:

$$\mathbf{z}_{n+1} = \mathbf{z}_n + \mathbf{v}_{n+1}, \tag{8}$$

where: $\mathbf{z}_0 \in \mathbb{R}^D$ is a starting position, $\mathbf{v}_0 = [0, 0, \dots, 0]$ is a starting velocity, $\mathbf{v}_{n+1}$ is the current velocity of particle ($\mathbf{v}_{n+1} = [v^1_{n+1}, v^2_{n+1}, \dots, v^D_{n+1}]$), $\mathbf{z}_n$ is the previous position of particle ($\mathbf{z}_n = [z^1_n, z^2_n, \dots, z^D_n]$). The algorithm sums the position of the particle $\mathbf{z}_n$ with its current velocity $\mathbf{v}_{n+1}$. The current velocity of the particle is determined by the inertia weight and the acceleration constants:

$$\mathbf{v}_{n+1} = \omega \mathbf{v}_n + \eta \mathbf{N}(\mathbf{z}_n), \tag{9}$$

where: $\mathbf{v}_n$—the previous velocity of particle, $\omega \in [0, 1)$—inertia weight, $\eta \in (0, 1]$—acceleration constant. Let us notice that (8) reduces to the classical Newton's method [25] if $\omega = 0$ and $\eta = 1$, and to the relaxed Newton's method [34] if $\omega = 0$ and $\eta \neq 1$. Moreover, we can observe that the proposed method joins the PSO algorithm and the Newton's method. The acceleration term in the PSO algorithm is responsible for pointing the particle in the direction of the best solution. In our method as the acceleration term we use the direction used in the Newton's method, which is responsible for moving the current point into direction of the solution. Thus, in each iteration we are moving towards the solution and eventually we will stop in the solution, because for $\omega < 1$ the particle slows down and $N(\mathbf{z}_n) = 0$, when we reach the solution.

The implementation of inertia weight (ω) and the acceleration constant (η) allows controlling particle dynamics in a wider range [4]. The selection of the value of these parameters is not deterministic. It depends on the problem being solved and the selected iteration method. These values are selected by tuning—it is a kind of art. The relationships between the ω and η parameters show a complex nature and the change of these parameters influences particle's dynamics, which can be visualised using algorithms described in Section 4.

A very important feature in numerical methods for solving systems of equations is the order of convergence of the method. For instance, the order of convergence of Newton's method is 2 [25]. In the case of the methods that are based on Newton's, Traub, Jarratt, etc. methods the order of convergence is proved (see for instance [8–14]), but in case of the methods that are based on metaheuristic algorithms, like the method presented in this paper, the order of convergence is not derived (see for instance [15–22]). This is due the fact that this order greatly varies for different values of the parameters used in the metaheuristic algorithms. We will show this on a very simple example. In this example to approximate the order of convergence we will use the computational order convergence introduced in [35], i.e.,

$$p \approx \frac{\ln\left(\|\mathbf{z}_{n+1} - \mathbf{z}_n\| / \|\mathbf{z}_n - \mathbf{z}_{n-1}\|\right)}{\ln\left(\|\mathbf{z}_n - \mathbf{z}_{n-1}\| / \|\mathbf{z}_{n-1} - \mathbf{z}_{n-2}\|\right)}. \tag{10}$$

Let us consider $f_1(x, y) = x^3 - 3xy^2 - 1$, $f_2(x, y) = 3x^2y - y^3$. We will solve system (4) by using (8) with various values of ω and η in (9) and three different starting points $\mathbf{z}_0$. The obtained results

are presented in Table 1. For $\omega = 0$ and $\eta = 1$ (classic Newton's method) we see that the method obtains second order of convergence, which is in accordance with the result proved in the literature. When we take a non-zero inertia, then we see that for a fixed starting point and varying ω and η we obtain very different values of the order. The same situation happens when we fix ω and η and change the starting points.

Table 1. Computational order of convergence for system given by $f_1(x,y) = x^3 - 3xy^2 - 1$, $f_2(x,y) = 3x^2y - y^3$ solved by (8).

Parameters (ω, η)	$\mathbf{z_0} = [-2.5, 2.5]$	$\mathbf{z_0} = [-2, -2]$	$\mathbf{z_0} = [2.5, 2.5]$
$(0.0, 1.0)$	2.06	2.01	2.03
$(0.1, 0.9)$	1.15	1.93	0.91
$(0.1, 0.8)$	1.10	0.61	0.31
$(0.1, 0.7)$	0.95	1.18	2.43
$(0.1, 0.6)$	1.28	1.30	1.34
$(0.2, 0.7)$	1.67	2.47	2.79
$(0.2, 0.6)$	0.79	1.96	1.18
$(0.25, 0.6)$	1.52	2.01	0.94
$(0.3, 0.6)$	0.54	1.68	2.82

Another problem studied in the context of solving systems of non-linear equation with iterative methods is the problem of strange (extraneous) fixed points, i.e., fixed points which are found by the method and are not the roots of the considered system [36]. Let us assume that (8) has converged to a solution. In such case the $\mathbf{v}_n$ in (9) is equal to 0. Therefore, method (8) behaves like relaxed Newton's method. As a result that relaxed Newton's method has no strange fixed point [25], the proposed method also has no strange fixed points.

3. Iteration Processes

Picard's iteration [37] is widely used in many iterative algorithms. This iteration can be described as follows:

$$\mathbf{z}_{n+1} = \mathbf{T}(\mathbf{z}_n). \tag{11}$$

Notice that (8) uses the Picard iteration, where $\mathbf{T} : \mathbb{R}^D \to \mathbb{R}^D$ takes the form

$$\mathbf{T}(\mathbf{z}_n) = \mathbf{z}_n + \mathbf{v}_{n+1}. \tag{12}$$

In the literature many other iteration processes can be found. Three most basic and commonly implemented algorithms of the approximate finding of fixed points of mappings are as follows:

1. The Mann iteration [38]:

$$\mathbf{z}_{n+1} = (1 - \alpha_n)\mathbf{z}_n + \alpha_n \mathbf{T}(\mathbf{z}_n), \ n = 0, 1, 2, \ldots, \tag{13}$$

where: $\alpha_n \in (0, 1]$ for all $n \in \mathbb{N}$, and the Mann iteration for $\alpha_n = 1$ reduces to the Picard iteration.

2. The Ishikawa iteration [39]:

$$\begin{aligned} \mathbf{z}_{n+1} &= (1 - \alpha_n)\mathbf{z}_n + \alpha_n \mathbf{T}(\mathbf{u}_n), \\ \mathbf{u}_n &= (1 - \beta_n)\mathbf{z}_n + \beta_n \mathbf{T}(\mathbf{z}_n), \ n = 0, 1, 2, \ldots, \end{aligned} \tag{14}$$

where: $\alpha_n \in (0, 1]$ and $\beta_n \in [0, 1]$ for all $n \in \mathbb{N}$, the Ishikawa iteration reduces to the Mann iteration when $\beta_n = 0$, and to the Picard iteration when $\alpha_n = 1$ and $\beta_n = 0$.

3. The Agarwal iteration [40] (*S*-iteration):

$$\begin{aligned} \mathbf{z}_{n+1} &= (1-\alpha_n)\mathbf{T}(\mathbf{z}_n) + \alpha_n \mathbf{T}(\mathbf{u}_n), \\ \mathbf{u}_n &= (1-\beta_n)\mathbf{z}_n + \beta_n \mathbf{T}(\mathbf{z}_n),\ n = 0,1,2,\ldots, \end{aligned} \tag{15}$$

where $\alpha_n \in [0,1]$ and $\beta_n \in [0,1]$ for all $n \in \mathbb{N}$, let us notice that the S-iteration reduces to the Picard iteration when $\alpha_n = 0$, or $\alpha_n = 1$ and $\beta_n = 0$.

A survey of various iteration processes and their dependencies can be found in [41].

An additional particle (a sample in the solution space) $\mathbf{u}_n$ (acting as a reference point) is applied in the Ishikawa and Agarwal iterations. In this case, each of these iterations performs two steps. The reference point $\mathbf{u}_n$ is set in the first step, and next a new particle position $\mathbf{z}_{n+1}$ is computed using the reference point. It allows better control of the particle's motion. In the paper a modification of the above-discussed iterations is proposed by applying the best position of the reference point and the best position of the particle—similar to the PSO algorithm.

The positions of the reference point $\mathbf{u}_n$ and the particle $\mathbf{z}_n$ are evaluated and compared to $\mathbf{u}_{best}$ and $\mathbf{z}_{best}$. In solutions space, the lower value of $\|\mathbf{F}(\mathbf{q})\|$ determines the better position of the reference point or the particle ($\mathbf{q}$). The next $\mathbf{u}_{best}$ and $\mathbf{z}_{best}$ are updated if they are worse than $\mathbf{u}_n$ and $\mathbf{z}_n$. The best position of the reference point $\mathbf{u}_{best}$ or position of the reference point $\mathbf{u}_n$ and the best position of the particle $\mathbf{z}_{best}$ or the position of the particle $\mathbf{z}_n$ (depending on the result of the evaluation) are used to determine the new position of the particle $\mathbf{z}_{n+1}$.

As a result of this modification, the next position of the particle $\mathbf{z}_{n+1}$ in the Equations (13)–(15) is determined by the following formulas:

1. The modified Mann iteration

$$\mathbf{z}_{n+1} = (1-\alpha_n)\mathbf{z}_{\{n|best\}} + \alpha_n \mathbf{T}(\mathbf{z}_n), \tag{16}$$

2. The modified Ishikawa iteration

$$\mathbf{z}_{n+1} = (1-\alpha_n)\mathbf{z}_{\{n|best\}} + \alpha_n \mathbf{T}(\mathbf{u}_{\{n|best\}}), \tag{17}$$

3. The modified Agarwal iteration

$$\mathbf{z}_{n+1} = (1-\alpha_n)\mathbf{T}(\mathbf{z}_{\{n|best\}}) + \alpha_n \mathbf{T}(\mathbf{u}_{\{n|best\}}). \tag{18}$$

All the presented iterations used only one mapping, but in the fixed point theory exist iterations that use several mappings and are applied to find common fixed points of the mappings. Examples of this type of iterations are the following:

1. The Das–Debata iteration [42]:

$$\begin{aligned} \mathbf{z}_{n+1} &= (1-\alpha_n)\mathbf{z}_n + \alpha_n \mathbf{T}_2(\mathbf{u}_n), \\ \mathbf{u}_n &= (1-\beta_n)\mathbf{z}_n + \beta_n \mathbf{T}_1(\mathbf{z}_n),\ n = 0,1,2,\ldots, \end{aligned} \tag{19}$$

where $\alpha_n \in (0,1]$ and $\beta_n \in [0,1]$ for all $n \in \mathbb{N}$, the Das–Debata iteration for $\mathbf{T}_1 = \mathbf{T}_2$ reduces to the Ishikawa iteration.

2. The Khan–Cho–Abbas iteration [43]:

$$\begin{aligned} \mathbf{z}_{n+1} &= (1-\alpha_n)\mathbf{T}_1(\mathbf{z}_n) + \alpha_n \mathbf{T}_2(\mathbf{u}_n), \\ \mathbf{u}_n &= (1-\beta_n)\mathbf{z}_n + \beta_n \mathbf{T}_1(\mathbf{z}_n),\ n = 0,1,2,\ldots, \end{aligned} \tag{20}$$

where $\alpha_n \in (0,1]$ and $\beta_n \in [0,1]$ for all $n \in \mathbb{N}$, let us notice that the Khan–Cho–Abbas iteration reduces to the Agarwal iteration when $\mathbf{T}_1 = \mathbf{T}_2$.

3. The generalised Agarwal's iteration [43]:

$$\begin{aligned} \mathbf{z}_{n+1} &= (1-\alpha_n)\mathbf{T}_3(\mathbf{z}_n) + \alpha_n \mathbf{T}_2(\mathbf{u}_n), \\ \mathbf{u}_n &= (1-\beta_n)\mathbf{z}_n + \beta_n \mathbf{T}_1(\mathbf{z}_n),\ n = 0,1,2,\ldots, \end{aligned} \tag{21}$$

where $\alpha_n \in (0,1]$ and $\beta_n \in [0,1]$ for all $n \in \mathbb{N}$, moreover the generalised Agarwal iteration reduces to the Khan–Cho–Abbas iteration when $\mathbf{T}_1 = \mathbf{T}_3$ and to the Agarwal iteration when $\mathbf{T}_1 = \mathbf{T}_2 = \mathbf{T}_3$.

Iterations described by Equations (19)–(21) can be also modified by introducing the best position of the reference point and the best position of the particle. As a result of this modification, equations describing $\mathbf{z}_{n+1}$ take the following form:

1. The modified Das–Debata iteration

$$\mathbf{z}_{n+1} = (1-\alpha_n)\mathbf{z}_{\{n|best\}} + \alpha_n \mathbf{T}_2(\mathbf{u}_{\{n|best\}}), \tag{22}$$

2. The modified Khan–Cho–Abbas iteration

$$\mathbf{z}_{n+1} = (1-\alpha_n)\mathbf{T}_1(\mathbf{z}_{\{n|best\}}) + \alpha_n \mathbf{T}_2(\mathbf{u}_{\{n|best\}}), \tag{23}$$

3. The modified generalised Agarwal iteration

$$\mathbf{z}_{n+1} = (1-\alpha_n)\mathbf{T}_3(\mathbf{z}_{\{n|best\}}) + \alpha_n \mathbf{T}_2(\mathbf{u}_{\{n|best\}}). \tag{24}$$

For all iterations $\mathbf{z}_n$ is used for the base type of iteration (Algorithm 1), $\mathbf{u}_{best}$ is used for the proposed algorithm with the best reference point (Algorithm 2), $\mathbf{z}_{best}$ is used for the proposed algorithm with the best particle position (Algorithm 3) and both $\mathbf{u}_{best}$ and $\mathbf{z}_{best}$ are used for the proposed Algorithm 4. The variety of different iteration processes is implemented in our algorithms. In the iterations we use (12) as the mapping with different values of ω and η parameters.

4. Visualisation of the Dynamics

A very similar method to the polynomiography [44] is used to visualise the dynamics of the proposed algorithms. The iteration methods presented in Section 3 are implemented in the algorithms and proper values of parameters are selected. ω, η for a single mapping $\mathbf{T}$ or $\omega_1, \omega_2, \omega_3$ and η_1, η_2, η_3 for $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3$ (depending on the chosen iteration). The maximum number of iterations m which algorithm should make, accuracy of the computations ε and a colouring function $C : \mathbb{N} \to \{0,1,\ldots,255\}^3$ are set. Then, each $\mathbf{z}_0$ in the solution space $\mathbf{A}$ is used in the algorithm. The iterations of the algorithm proceed till the convergence criterion:

$$\|\mathbf{z}_{n+1} - \mathbf{z}_n\| < \varepsilon \tag{25}$$

or the maximum number of iterations is reached. A colour corresponding to the performed number of iterations is assigned to $\mathbf{z}_0$ using colouring function C. Iterations described in Section 3 and their modifications using the best particle position and the best reference point position allow proposing five algorithms. Algorithm 1 (denoted in the text by I) is the base algorithm, without modification. Algorithm 2 (denoted in the article by II) performs the selection of the best position of the reference point. Algorithm 3 (denoted by III) performs the selection of the best particle. Algorithm 4 (denoted by IV) combines the modifications introduced in Algorithms 2 and 3. Algorithm 5 is a modification of Algorithm 3 because Mann iteration does not use a reference point and it is denoted by $IIIM$.

Algorithm 1: Visualisation of the dynamics—the base Algorithm (I)

Input: **F**—function, $\mathbf{A} \subset \mathbb{R}^D$—solution space, m—the maximum number of iterations, I_q^I—iteration method, $q \in [0,1]^N$—parameters of the iteration I_q, ω, ω_1, ω_2, ω_3, η, η_1, η_1, η_2, η_3—parameters defining functions T, T_1, T_2, T_3, C—colouring function, ε—accuracy

Output: visualisation of the dynamics

foreach $\mathbf{z}_0 \in \mathbf{A}$ **do**
  $i = 0$
  $\mathbf{v}_0 = [0, 0, \ldots, 0]$
  while $i \leq m$ **do**
    $\mathbf{z}_{n+1} = I_q^I(\mathbf{z}_n)$
    if $\|\mathbf{z}_{n+1} - \mathbf{z}_n\| < \varepsilon$ **then**
      break
    $i = i + 1$
  colour $\mathbf{z}_0$ with $C(i)$

Algorithm 2: Visualisation of the dynamics with the best reference point (II)

Input: **F**—function, $\mathbf{A} \subset \mathbb{R}^D$—solution space, m—the maximum number of iterations, I_q^r—part of the iteration method computing the reference point $\mathbf{u}_n$, I_q^{II}—part of the iteration method computing the particle $\mathbf{z}_{n+1}$ using the best position of the reference point $\mathbf{u}_{best}$, $q \in [0,1]^N$—parameters of the iteration, ω, ω_1, ω_2, ω_3, η, η_1, η_1, η_2, η_3—parameters defining functions $\mathbf{T}$, $\mathbf{T}_1$, $\mathbf{T}_2$, $\mathbf{T}_3$, C—colouring function, ε—accuracy

Output: visualisation of the dynamics

foreach $\mathbf{z}_0 \in \mathbf{A}$ **do**
  $i = 0$
  $\mathbf{v}_0 = [0, 0, \ldots, 0]$
  $\mathbf{u}_{best} = \mathbf{z}_0$
  while $i \leq m$ **do**
    $\mathbf{u}_n = I_q^r(\mathbf{z}_n)$
    if $\|\mathbf{F}(\mathbf{u}_n)\| < \|\mathbf{F}(\mathbf{u}_{best})\|$ **then**
      $\mathbf{u}_{best} = \mathbf{u}_n$
    $\mathbf{z}_{n+1} = I_q^{II}(\mathbf{z}_n, \mathbf{u}_{best})$
    if $\|\mathbf{z}_{n+1} - \mathbf{z}_n\| < \varepsilon$ **then**
      break
    $i = i + 1$
  colour $\mathbf{z}_0$ with $C(i)$

Algorithm 3: Visualisation of the dynamics with the best particle (III)

Input: $\mathbf{F}$—function, $\mathbf{A} \subset \mathbb{R}^D$—solution space, m—the maximum number of iterations, I_q^r—part of the iteration method computing the reference point $\mathbf{u}_n$, I_q^{III}—part of the iteration method computing the particle $\mathbf{z}_{n+1}$ using the best position of the particle $\mathbf{z}_{best}$, $q \in [0,1]^N$—parameters of the iteration, ω, ω_1, ω_2, ω_3, η, η_1, η_1, η_2, η_3—parameters defining functions $\mathbf{T}$, $\mathbf{T}_1$, $\mathbf{T}_2$, $\mathbf{T}_3$, C—colouring function, ε—accuracy

Output: visualisation of the dynamics

foreach $\mathbf{z}_0 \in \mathbf{A}$ **do**
- $i = 0$
- $\mathbf{v}_0 = [0, 0, \ldots, 0]$
- $\mathbf{z}_{best} = \mathbf{z}_0$
- **while** $i \leq m$ **do**
  - $\mathbf{u}_n = I_q^r(\mathbf{z}_n)$
  - $\mathbf{z}_{n+1} = I_q^{III}(\mathbf{z}_{best}, \mathbf{z}_n)$
  - **if** $\|\mathbf{z}_{n+1} - \mathbf{z}_n\| < \varepsilon$ **then**
    - **break**
  - **if** $\|\mathbf{F}(\mathbf{z}_{n+1})\| < \|\mathbf{F}(\mathbf{z}_{best})\|$ **then**
    - $\mathbf{z}_{best} = \mathbf{z}_{n+1}$
  - $i = i + 1$
- colour $\mathbf{z}_0$ with $C(i)$

Algorithm 4: Visualisation of the dynamics with both the best reference point and particle (IV)

Input: $\mathbf{F}$—function, $\mathbf{A} \subset \mathbb{R}^D$—solution space, m—the maximum number of iterations, I_q^{IV}—part of the iteration method computing the particle $\mathbf{z}_{n+1}$ using the best position of the reference point $\mathbf{u}_{best}$ and the best position of the particle $\mathbf{z}_{best}$, $q \in [0,1]^N$—parameters of the iteration, ω, ω_1, ω_2, ω_3, η, η_1, η_1, η_2, η_3—parameters defining functions $\mathbf{T}$, $\mathbf{T}_1$, $\mathbf{T}_2$, $\mathbf{T}_3$, C—colouring function, ε—accuracy

Output: visualisation of the dynamics

foreach $\mathbf{z}_0 \in \mathbf{A}$ **do**
- $i = 0$
- $\mathbf{v}_0 = [0, 0, \ldots, 0]$
- $\mathbf{u}_{best} = \mathbf{z}_0$
- $\mathbf{z}_{best} = \mathbf{z}_0$
- **while** $i \leq m$ **do**
  - $\mathbf{u}_n = I_q^r(\mathbf{z}_n)$
  - **if** $\|\mathbf{F}(\mathbf{u}_n)\| < \|\mathbf{F}(\mathbf{u}_{best})\|$ **then**
    - $\mathbf{u}_{best} = \mathbf{u}_n$
  - $\mathbf{z}_{n+1} = I_q^{IV}(\mathbf{z}_{best}, \mathbf{z}_n, \mathbf{u}_{best})$
  - **if** $\|\mathbf{z}_{n+1} - \mathbf{z}_n\| < \varepsilon$ **then**
    - **break**
  - **if** $\|\mathbf{F}(\mathbf{z}_{n+1})\| < \|\mathbf{F}(\mathbf{z}_{best})\|$ **then**
    - $\mathbf{z}_{best} = \mathbf{z}_{n+1}$
  - $i = i + 1$
- colour $\mathbf{z}_0$ with $C(i)$

Algorithm 5: Visualisation of the dynamics with the best particle for the Mann iteration $(IIIM)$

Input: **F**—function, $\mathbf{A} \subset \mathbb{R}^D$—solution space, m—the maximum number of iterations, I_q^{IIIM}—part of the iteration method computing the particle $\mathbf{z}_{n+1}$ using the best position of the particle $\mathbf{z}_{best}$, $q \in [0,1]^N$—parameters of the iteration, ω, ω_1, ω_2, ω_3, η, η_1, η_1, η_2, η_3—parameters defining functions $\mathbf{T}$, $\mathbf{T}_1$, $\mathbf{T}_2$, $\mathbf{T}_3$, C—colouring function, ε—accuracy

Output: visualisation of the dynamics

```
foreach z_0 ∈ A do
    i = 0
    v_0 = [0, 0, ..., 0]
    z_best = z_0
    while i ≤ m do
        z_{n+1} = I_q^{IIIM}(z_best, z_n)
        if ||z_{n+1} − z_n|| < ε then
            break
        if ||F(z_{n+1})|| < ||F(z_best)|| then
            z_best = z_{n+1}
        i = i + 1
    colour z_0 with C(i)
```

The iteration method is denoted by I_q^a for a given algorithm, where q is a vector of parameters of the iteration and a is one of the algorithms $\{I|II|III|IIIM|IV\}$ or r—it is the part of iteration determining position of the reference point.

The domain (solution space) **A** is defined in a D-dimensional space, thus the algorithms return polynomiographs in this space. For $D = 2$ a single image is obtained or for $D > 2$ a two-dimensional cross-section of **A** can be selected for visualisation.

5. Discussion on the Research Results

In this section we present and discuss the obtained results of visualising the dynamics of the method introduced in Section 2 together with the various iteration methods presented in Section 3 using algorithms described in Section 4. Let $\mathbb{C}$ be the field of complex numbers with a complex number $c = x + iy$ where $i = \sqrt{-1}$ and $x, y \in \mathbb{R}$. In the experiments we want to solve the following non-linear equation

$$p(c) = 0 \tag{26}$$

where $p(c) = c^3 - 1$.

This equation can be written in the following form:

$$0 = c^3 - 1 = (x + iy)^3 - 1 = x^3 - 3xy^2 - 1 + (3x^2y - y^3)i. \tag{27}$$

Now, Equation (27) can be transformed into a system of two equations with two variables, i.e.,

$$\mathbf{F}(x, y) = \begin{bmatrix} f_1(x, y) \\ f_2(x, y) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \mathbf{0}, \tag{28}$$

where $f_1(x, y) = x^3 - 3xy^2 - 1$, $f_2(x, y) = 3x^2y - y^3$. The set of solutions of this system is the following: $[1, 0]$, $[-0.5, -0.866025]$, $[-0.5, 0.866025]$ and $\mathbf{A} = [-2.0, 2.0]^2$ ($[-0.5, 0.5]^2$ is used for magnification of the centre parts of polynomiographs).

Moreover, several additional test functions are used:

$$\begin{aligned} 0 = c^4 - 10c^2 + 9 &= (x+iy)^4 - 10(x+iy)^2 + 9 \\ &= x^4 - 6x^2y^2 + y^4 - 10x^2 + 10y^2 + 9 + (4x^3y - 4xy^3 - 20xy)i, \end{aligned} \tag{29}$$

where: $f_1(x,y) = x^4 - 6x^2y^2 + y^4 - 10x^2 + 10y^2 + 9$, $f_2(x,y) = 4x^3y - 4xy^3 - 20xy$ and the set of solutions of this system is the following: $[-3.0, 0.0]$, $[-1.0, 0.0]$, $[1.0, 0.0]$, $[3.0, 0.0]$ and $\mathbf{A} = [-4.0, 4.0] \times [-2.0, 2.0]$;

$$\begin{aligned} 0 = c^5 - c &= (x+iy)^5 - (x+iy) \\ &= x^5 - 10x^3y^2 + 5xy^4 - x + (5x^4y - 10x^2y^3 + y^5 - y)i, \end{aligned} \tag{30}$$

where: $f_1(x,y) = x^5 - 10x^3y^2 + 5xy^4 - x$, $f_2(x,y) = 5x^4y - 10x^2y^3 + y^5 - y$ and the set of solutions of this system is the following: $[-1.0, 0.0]$, $[0.0, -1.0]$, $[0.0, 0.0]$, $[0.0, 1.0]$, $[1.0, 0.0]$ and $\mathbf{A} = [-2.0, 2.0]^2$;

$$\begin{aligned} 0 = c^6 + 10c^3 - 8 &= (x+iy)^6 + 10(x+iy)^3 - 8 \\ &= x^6 - 15x^4y^2 + 15x^2y^4 - y^6 + 10x^3 - 30xy^2 - 8 + (6x^5y - 20x^3y^3 + 6xy^5 + 30x^2y - 10y^3)i, \end{aligned} \tag{31}$$

where: $f_1(x,y) = x^6 - 15x^4y^2 + 15x^2y^4 - y^6 + 10x^3 - 30xy^2 - 8$, $f_2(x,y) = 6x^5y - 20x^3y^3 + 6xy^5 + 30x^2y - 10y^3$ and the set of solutions of this system is the following (approximately):$[-2.207, 0]$, $[-0.453, -0.785]$, $[-0.453, 0.785]$, $[0.906, 0]$, $[1.103, -1.911]$, $[1.103, 1.911]$ and $\mathbf{A} = [-2.3, 1.7] \times [-2.0, 2.0]$.

In the experiments, to solve (28) we use the method introduced in Section 2. The same colour map (Figure 1) was used to colour all the obtained polynomiographs. Furthermore, in every experiment we used the following common parameters: $m = 256$, $\varepsilon = 0.01$, image resolution 800×800 pixels.

Figure 1. Colour map used in the experiments.

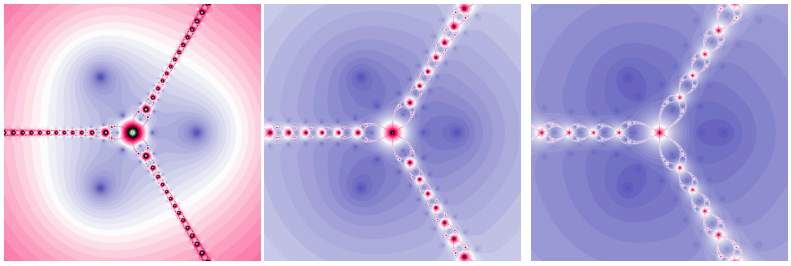
The algorithms used in the experiments were implemented in the C++ programming language. The experiments were conducted on a computer with the Intel Core i7 processor, 8 GB RAM and Linux Ubuntu 18.04 LTS.

5.1. The Picard Iteration

The motion of particles depends on the acceleration constant (η) and inertia weight (ω). These parameters control dynamics of a particle behaviour. The polynomiograph visualises particle dynamics and allows analysing parameters impact on the algorithm's operation. Polynomiographs are created using Algorithm 1. The dynamics visualisations for the Picard iteration using $\omega = 0.0$ and varying η are presented in Figure 2. Earlier, in Section 2, we noticed that the proposed method reduces to the relaxed Newton's method for $\omega = 0$ and $\eta \neq 1$ and to the Newton's method for $\omega = 0$ and $\eta = 1$, so Figure 2a–e present polynomiographs for the relaxed Newton's method, whereas Figure 2f for the classical Newton's method. The polynomiograph generation times are also given in Figure 2. The time decreases as the dynamics controlled by the acceleration constant increases.

Polynomiographs of the Picard iteration for low value of acceleration constant ($\eta = 0.1$) and varying ω are presented in Figure 3. The increase in the inertia weight results in the increase in particle dynamics. Both too small and too high particle dynamics cause the increase in the polynomiographs creation time. The shortest time was obtained for Figure 3b. Every change of dynamics creates a new image.

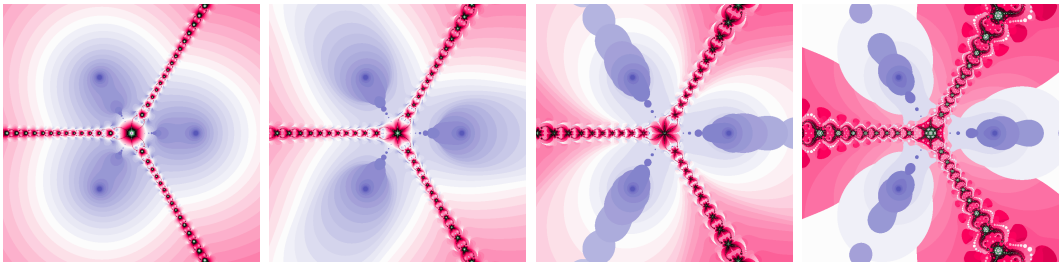
The acceleration constant is equal to 0.3 for images in Figure 4. Change in particle's dynamics is realised by the increase of the inertia weight. As in the previous example, the proper selection of particle's dynamics minimises the creation time of the polynomiograph (see Figure 4b). The increase in the particle dynamics is expressively illustrated by polynomiographs.



(a) $\eta = 0.2$, $t = 0.62s$, (I) **(b)** $\eta = 0.4$, $t = 0.33s$, (I) **(c)** $\eta = 0.6$, $t = 0.25s$, (I)

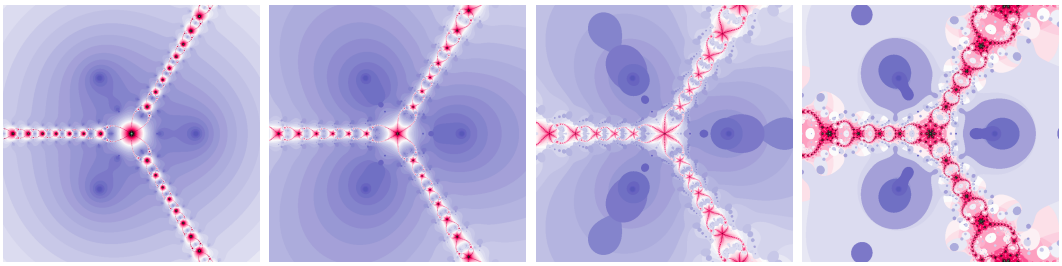
(d) $\eta = 0.8$, $t = 0.21s$, (I) **(e)** $\eta = 0.9$, $t = 0.20s$, (I) **(f)** $\eta = 1.0$, $t = 0.19s$, (I)

Figure 2. Polynomiographs of the Picard iteration for $\omega = 0.0$ and varying η.



(a) $\omega = 0.5$, $t = 0.59s$, (I) **(b)** $\omega = 0.6$, $t = 0.58s$, (I) **(c)** $\omega = 0.7$, $t = 0.64s$, (I) **(d)** $\omega = 0.8$, $t = 0.75s$, (I)

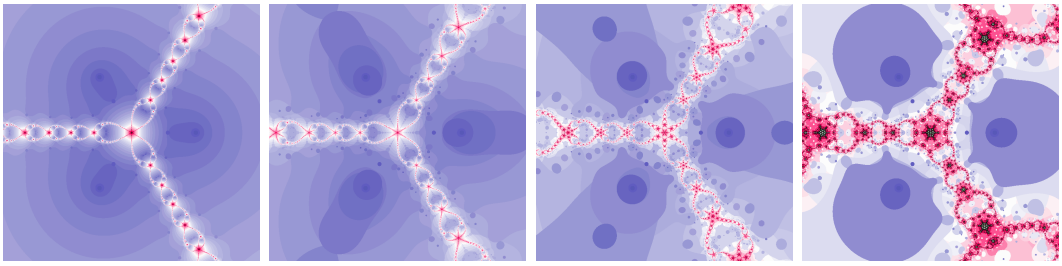
Figure 3. Polynomiographs of the Picard iteration for $\eta = 0.1$ and varying ω.



(a) $\omega = 0.1$, $t = 0.44s$, (I) **(b)** $\omega = 0.3$, $t = 0.33s$, (I) **(c)** $\omega = 0.5$, $t = 0.38s$, (I) **(d)** $\omega = 0.7$, $t = 0.53s$, (I)

Figure 4. Polynomiographs of the Picard iteration for $\eta = 0.3$ and varying ω.

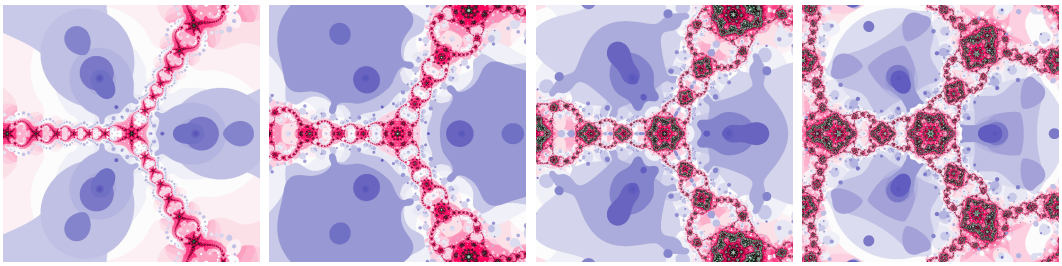
In Figure 5 we see polynomiographs generated using the same values of the inertia weight as in Figure 4, but with higher value of the acceleration constant—$\eta = 0.5$. The increase in the inertia weight causes the increase of the time of image creation. It is the consequence of the excessive growth of particle's dynamics.



(**a**) $\omega = 0.1$, $t = 0.25s$, (I) (**b**) $\omega = 0.3$, $t = 0.28s$, (I) (**c**) $\omega = 0.5$, $t = 0.35s$, (I) (**d**) $\omega = 0.7$, $t = 0.49s$, (I)

Figure 5. Polynomiographs of the Picard iteration for $\eta = 0.5$ and varying ω.

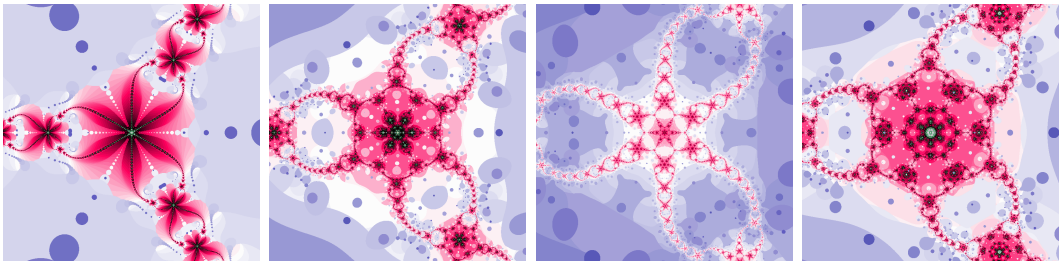
Polynomiographs of the Picard iteration for a constant value of the inertia weight ($\omega = 0.7$) and varying acceleration constant are presented in Figure 6. For the images in Figure 6c,d the dynamics presented by the polynomiographs is high (the parts of the image are blurred—there are non-contrast patterns created by the points). Moreover, let us notice that the generation time for Figure 6a is lower than the time obtained for Figure 2a (0.62s—the Picard iteration with $\omega = 0$, $\eta = 0.2$). Thus, by increasing the inertia we were able to obtain a shorter time than in the case of the relaxed Newton's method with the same value of acceleration.



(**a**) $\eta = 0.2$, $t = 0.54s$, (I) (**b**) $\eta = 0.4$, $t = 0.48s$, (I) (**c**) $\eta = 0.6$, $t = 0.63s$, (I) (**d**) $\eta = 0.8$, $t = 0.74s$, (I)

Figure 6. Polynomiographs of the Picard iteration for $\omega = 0.7$ and varying η.

The magnifications of the central part of selected polynomiographs of the Picard iteration are presented in Figure 7. Changes in particle dynamics paint beautiful patterns. They can be an inspiration for creating mosaics.



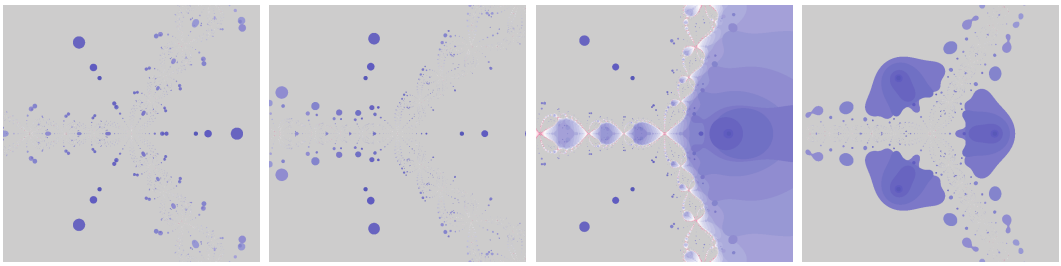
(**a**) Figure 3c (**b**) Figure 4d (**c**) Figure 5c (**d**) Figure 6b

Figure 7. Magnification of the central part of selected polynomiographs of the Picard iteration.

5.2. The Mann Iteration

Su and Qin in [45] discussed a strong convergence of iterations in the case of a fixed reference point. We start the examples for the Mann iteration with an example showing the behaviour of the Mann iteration implementing a fixed reference point. The results are presented in Figure 8.

For polynomiograph presented in Figure 8a the fixed reference point is in its centre $[0,0]$, whereas in Figure 8b this point is in the position $[-1,0]$—the behaviour of the algorithm is similar in both cases. For the polynomiograph in Figure 8c the fixed reference point is placed in the root position $[1,0]$ (in the best position). It causes a significant improvement in the algorithm operation near the root position. For polynomiograph presented in Figure 8d, a fixed reference point is the starting point—it results in greater efficiency in the algorithm operation near positions of the roots. A better position of the reference point can significantly improve the algorithm operation—however, it is impossible to determine its best position. In the proposed methods, instead of using a fixed reference point we use the locally best position of the reference point. This position is modified when the new position of the reference point is better than the previous one.

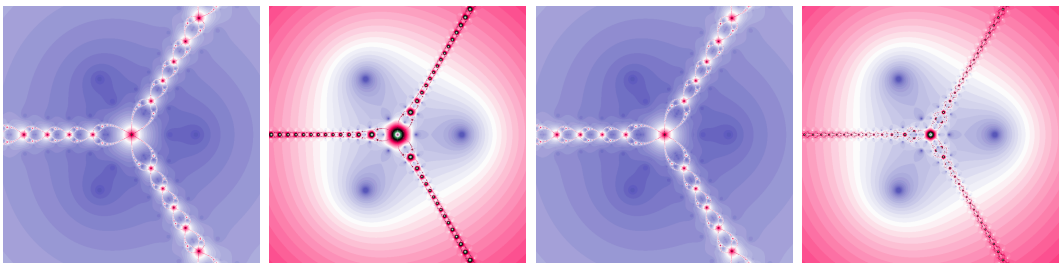


(**a**) $t = 8.45s$ (**b**) $t = 8.43s$ (**c**) $t = 5.02s$ (**d**) $t = 7.15s$

Figure 8. Polynomiographs of the Mann iteration with the fixed reference point for $\alpha = 0.9$, $\omega = 0.3$, $\eta = 0.5$. The positions of the reference point: (**a**) $[0,0]$, (**b**) $[-1,0]$, (**c**) $[1,0]$ (one of the roots), (**d**) the starting point.

The Mann iteration allows to compare the operation of algorithm I with the Algorithm $IIIM$. The choice of the best position of the particle will cause the change in the dynamics, it can result in a better convergence of the particle—this mechanics is used in many algorithms. Visualisation is a great tool for presenting this mechanics of algorithm operation.

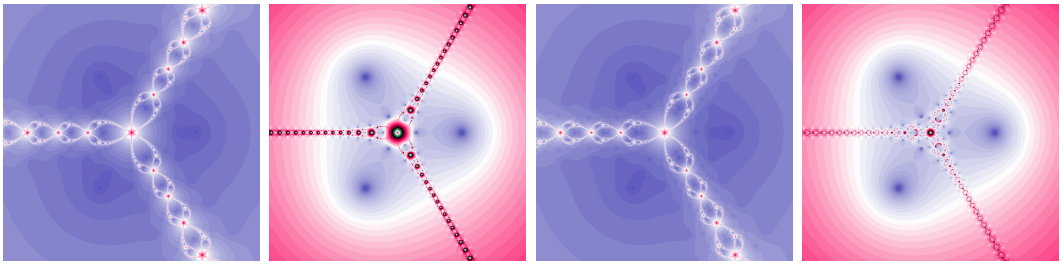
Polynomiographs of the Mann iteration for $\omega = 0.0$, $\eta = 0.6$ and varying α are presented in Figure 9. A small value of α causes a reduction in particle's dynamics. When Algorithm $IIIM$ is used, the areas with the higher dynamics (areas of dynamically changing pattern) are narrowed (compare Figure 9a with Figure 9c).



(**a**) $\alpha = 0.9$, $t = 0.27s$, (I) (**b**) $\alpha = 0.3$, $t = 0.73s$, (I) (**c**) $\alpha = 0.9$, $t = 0.32s$, $(IIIM)$ (**d**) $\alpha = 0.3$, $t = 0.82s$, $(IIIM)$

Figure 9. Polynomiographs of the Mann iteration for $\omega = 0.0$ and $\eta = 0.6$ varying α.

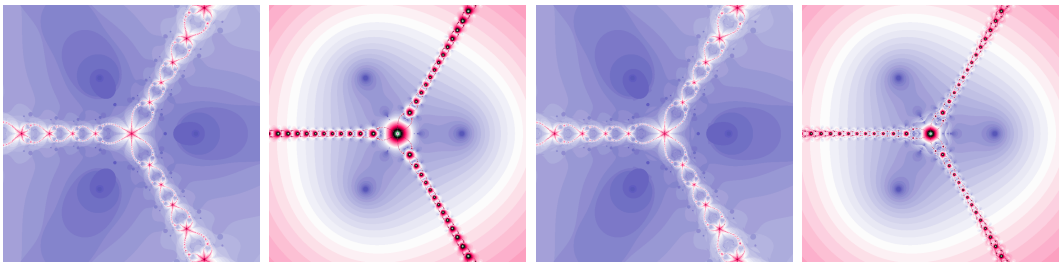
The particle dynamics for polynomiographs in Figure 10 is greater than in Figure 9. The high value of the α coefficient limits the effect of selecting the best particle's position in Algorithm $IIIM$. Nevertheless, in Figure 10c we observe a narrowing of areas of high dynamics and small changes in the shape of neighbouring areas. Limiting the particle's dynamics by decreasing in the value of the α coefficient affects the visualisation of the influence of the best particle position on the Algorithm $IIIM$ operation (compare Figure 10b and Figure 10d)—similarly to Figure 9d.



(a) $\alpha = 0.8$, $t = 0.23s$, (I) **(b)** $\alpha = 0.2$, $t = 0.77s$, (I) **(c)** $\alpha = 0.8$, $t = 0.25s$, $(IIIM)$ **(d)** $\alpha = 0.2$, $t = 0.81s$, $(IIIM)$

Figure 10. Polynomiographs of the Mann iteration for $\omega = 0.0$, $\eta = 0.9$ and varying α.

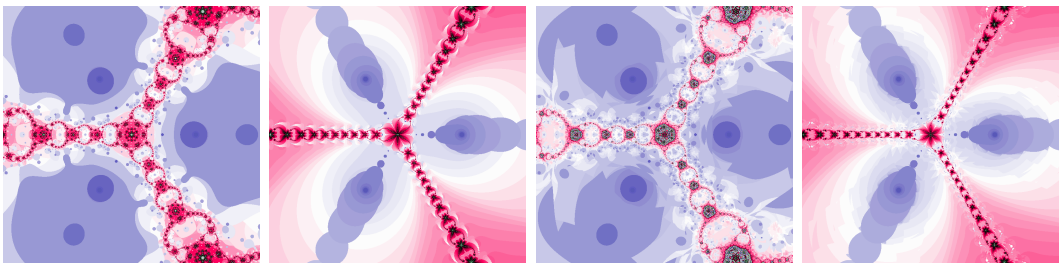
In the examples presented in Figure 11, the increase in particle dynamics is obtained by including inertia. The increase in particle dynamics is evident in polynomiographs in Figure 11a,c. As in the previous cases, Algorithm $IIIM$ makes changes visible only in areas with higher dynamics.



(a) $\alpha = 0.9$, $t = 0.29s$, (I) **(b)** $\alpha = 0.3$, $t = 0.58s$, (I) **(c)** $\alpha = 0.9$, $t = 0.33s$, $(IIIM)$ **(d)** $\alpha = 0.3$, $t = 0.66s$, $(IIIM)$

Figure 11. Polynomiographs of the Mann iteration for $\omega = 0.3$, $\eta = 0.5$ and varying α.

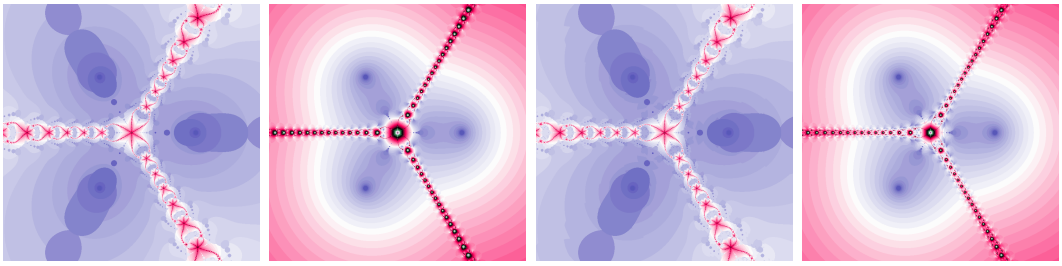
Another increase in the inertia weight ($\omega = 0.7$) causes the increase in dynamics shown in images in Figure 12a,b. The changes in dynamics caused by the Algorithm $IIIM$ operation are visible in the whole area of polynomiograph (see Figure 12c,d). The time of creation of polynomiograph in Figure 12d, when compared to Figure 12b, is shorter. The proposed modification of the algorithm can accelerate finding the solution.



(a) $\alpha = 0.8$, $t = 0.50s$, (I) **(b)** $\alpha = 0.2$, $t = 0.66s$, (I) **(c)** $\alpha = 0.8$, $t = 0.63s$, $(IIIM)$ **(d)** $\alpha = 0.2$, $t = 0.78s$, $(IIIM)$

Figure 12. Polynomiographs of the Mann iteration for $\omega = 0.7$, $\eta = 0.5$ and varying α.

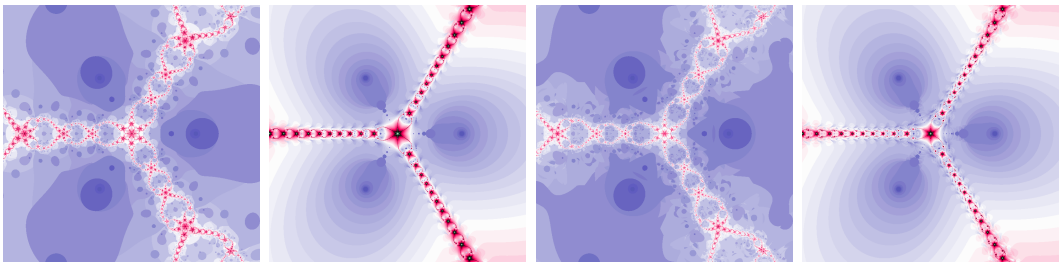
In polynomiographs in Figure 13, the particle dynamics are limited by the small value of η ($\eta = 0.3$). The changes in dynamics caused by the Algorithm $IIIM$ are in areas of high dynamics, the neighbourhood of these areas and places where the particle gets high velocity (the long distance from the centre of the image). This is clearly visible in images in Figure 13c,d.



(**a**) $\alpha = 0.9$, $t =$ $0.40s$, (I) (**b**) $\alpha = 0.3$, $t =$ $0.68s$, (I) (**c**) $\alpha = 0.9$, $t =$ $0.47s$, $(IIIM)$ (**d**) $\alpha = 0.3$, $t =$ $0.77s$, $(IIIM)$

Figure 13. Polynomiographs of the Mann iteration for $\omega = 0.5$, $\eta = 0.3$ and varying α.

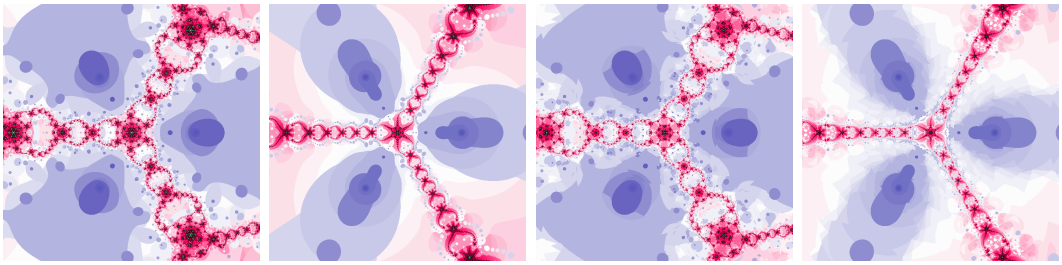
Polynomiographs presented in Figure 14 show the increase in dynamics due to the increase in the acceleration constant ($\eta = 0.7$). The dynamics is high for image in Figure 14a, and in Figure 14b the dynamics is limited by the small value of α. Algorithm $IIIM$ clearly influences the change in particle's dynamics—Figure 14c. The dynamics caused by the Algorithm $IIIM$ operation are much smaller in Figure 14d because the dynamics are limited by the small value of α.



(**a**) $\alpha = 0.8$, $t =$ $0.35s$, (I) (**b**) $\alpha = 0.2$, $t =$ $0.50s$, (I) (**c**) $\alpha = 0.8$, $t =$ $0.38s$, $(IIIM)$ (**d**) $\alpha = 0.2$, $t =$ $0.57s$, $(IIIM)$

Figure 14. Polynomiographs of the Mann iteration for $\omega = 0.5$, $\eta = 0.7$ and varying α.

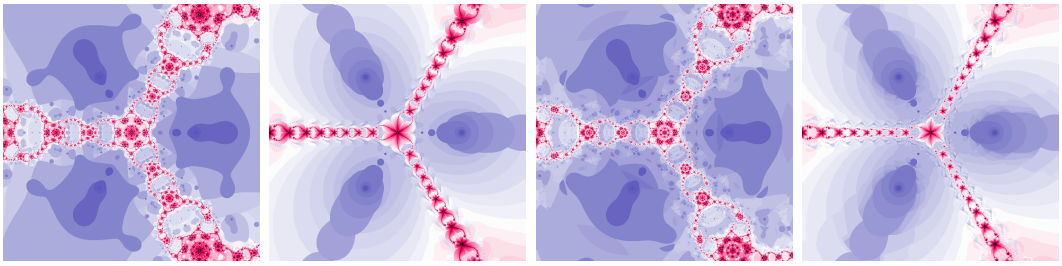
The inertia weight has much greater effect on particle dynamics than the acceleration constant. The dynamics presented in Figure 15 is high due to the high value of ω and η. The dynamics changes, caused by the best position of the particle, are clearly visible.



(**a**) $\alpha = 0.9$, $t =$ $0.53s$, (I) (**b**) $\alpha = 0.3$, $t =$ $0.58s$, (I) (**c**) $\alpha = 0.9$, $t =$ $0.61s$, $(IIIM)$ (**d**) $\alpha = 0.3$, $t =$ $0.69s$, $(IIIM)$

Figure 15. Polynomiographs of the Mann iteration for $\omega = 0.7$, $\eta = 0.6$ and varying α.

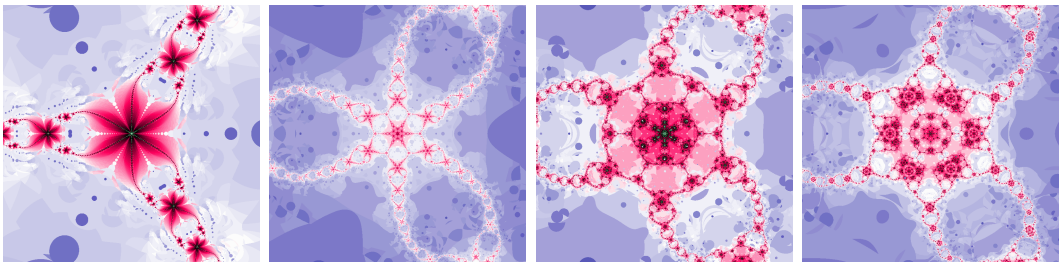
Similarly, as in the example presented in Figure 15, the dynamics in Figure 16 are shaped by the high value of ω and η ($\omega = 0.6$, $\eta = 0.8$). Additionally, in this case the dynamics paints some interesting images.



(**a**) $\alpha = 0.8$, $t = 0.42s$, (I) (**b**) $\alpha = 0.2$, $t = 0.52s$, (I) (**c**) $\alpha = 0.8$, $t = 0.46s$, $(IIIM)$ (**d**) $\alpha = 0.2$, $t = 0.60s$, $(IIIM)$

Figure 16. Polynomiographs of the Mann iteration for $\omega = 0.6$, $\eta = 0.8$ and varying α.

The magnifications of the central part of selected polynomiographs of the Mann iteration and its modifications are presented in Figure 17. Changes in the particle dynamics are visualised by colourful mosaics, which can be used in design to create ornaments.



(**a**) Figure 12d (**b**) Figure 14c (**c**) Figure 15c (**d**) Figure 16c

Figure 17. Magnification of the central part of selected polynomiographs of the Mann iteration.

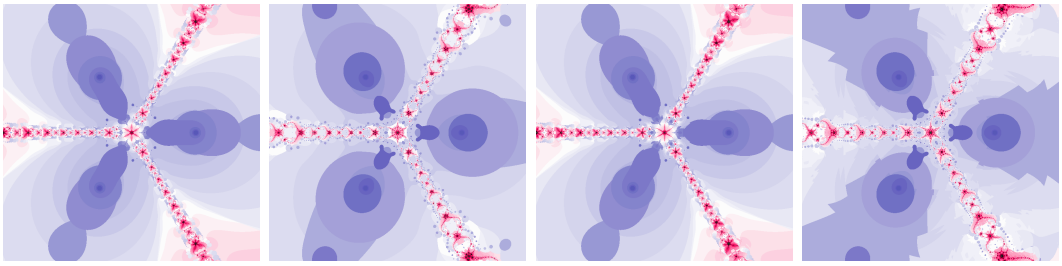
5.3. The Ishikawa and the Das–Debata Iterations

Ishikawa and Das–Debata iterations use a reference point to determine a new particle position. In the presented analysis, base Algorithm I and its three modifications are used: Algorithm II using the best position of the reference point, the Algorithm III using the best position of the particle, and both these modifications in the Algorithm IV.

Figure 18 shows polynomiographs of all four algorithms for two values of α ($\alpha = 0.4$ and $\alpha = 0.8$). The low α value limits the influence of the reference point, so images in Figure 18a and Figure 18c are very similar. Algorithm III using the best particle position has a significant influence on the form of the polynomiograph. It results in a strong similarity between images in Figure 18e and Figure 18g. The dynamics shown in Figure 18b is high. The influence of the best reference point is shown in Figure 18d because of the large value of α. Changes in dynamics are also visible in Figure 18f created by the algorithm using the best position of particle. Each of the algorithms has a different influence on the dynamics of the particle. These features are combined in Algorithm IV—it clearly visualises the image in Figure 18h.

A low value of β reduces the impact of the transformation operator on the dynamics of reference point creation. It has the same effect as the low value of α—for this reason Figure 19a is similar to Figure 19c,e–g. The increase in the β coefficient results in obtaining different features of the images in Figure 19b,d,f,h—the image in Figure 19h combines the features of the images in Figure 19d,f.

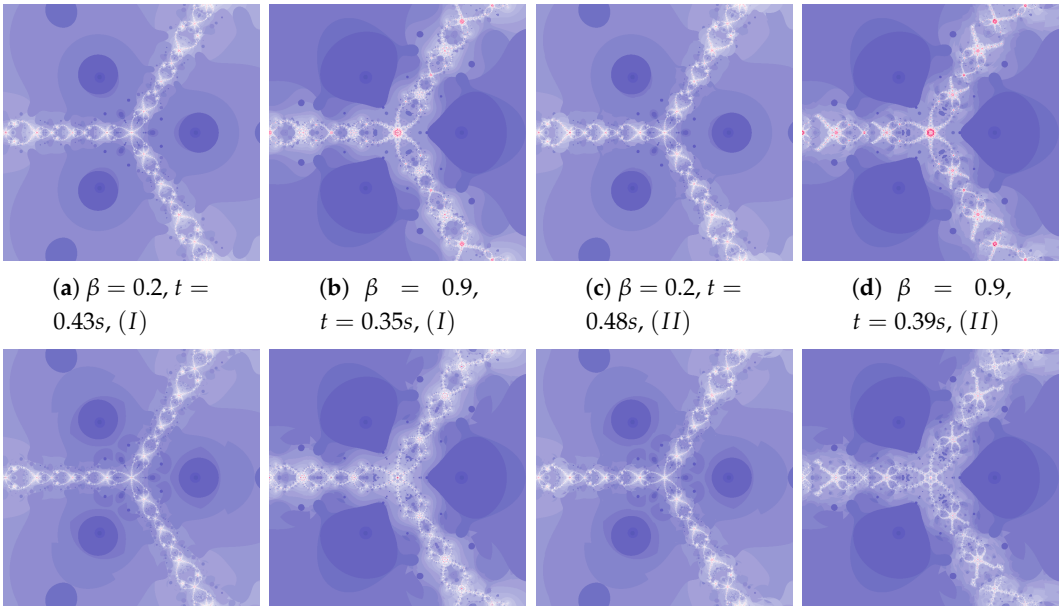
The dynamics presented by the polynomiographs depends on the dynamics of the creation of the reference point and its processing. The parameters ω_1 and η_1 are responsible for the dynamics of the reference point creation and the parameters ω_2 and η_2 are responsible for its processing. The change in these parameters allows to obtain the effects described above.



(a) $\alpha = 0.4$, $t = 0.83s$, (I) **(b)** $\alpha = 0.8$, $t = 0.76s$, (I) **(c)** $\alpha = 0.4$, $t = 0.92s$, (II) **(d)** $\alpha = 0.8$, $t = 0.87s$, (II)

(e) $\alpha = 0.4$, $t = 0.89s$, (III) **(f)** $\alpha = 0.8$, $t = 0.81s$, (III) **(g)** $\alpha = 0.4$, $t = 0.99s$, (IV) **(h)** $\alpha = 0.8$, $t = 0.94s$, (IV)

Figure 18. Polynomiographs of the Ishikawa iteration for $\beta = 0.4$, $\omega_1 = 0.7$, $\eta_1 = 0.3$, $\omega_2 = 0.7$, $\eta_2 = 0.3$ and varying α.

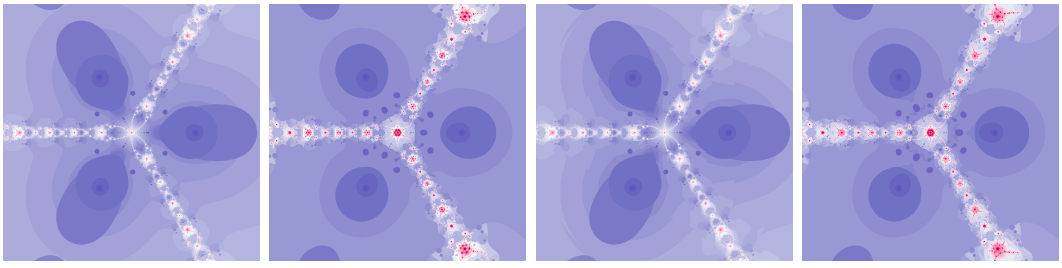


(a) $\beta = 0.2$, $t = 0.43s$, (I) **(b)** $\beta = 0.9$, $t = 0.35s$, (I) **(c)** $\beta = 0.2$, $t = 0.48s$, (II) **(d)** $\beta = 0.9$, $t = 0.39s$, (II)

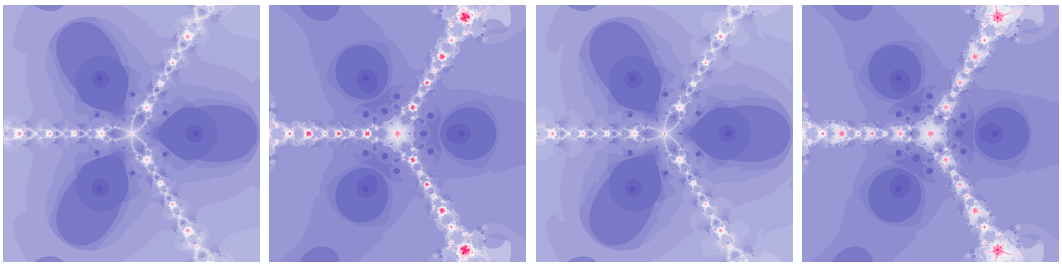
(e) $\beta = 0.2$, $t = 0.46s$, (III) **(f)** $\beta = 0.9$, $t = 0.39s$, (III) **(g)** $\beta = 0.2$, $t = 0.51s$, (IV) **(h)** $\beta = 0.9$, $t = 0.43s$, (IV)

Figure 19. Polynomiographs of the Ishikawa iteration for $\alpha = 0.7$, $\omega_1 = 0.4$, $\eta_1 = 0.8$, $\omega_2 = 0.4$, $\eta_2 = 0.8$ and varying β.

Polynomiographs of the Das–Debata iteration for varying ω_1 are presented in Figure 20 and for varying η_1 in Figure 21. Polynomiographs created by Algorithms I and II are similar—it means that the influence of choosing the best reference point is small for such selected parameters of the algorithm. The influence of the best position of particle has a significant effect on the form of the polynomiographs created by the Algorithm III—these polynomiographs have many common features with the polynomiographs created by the Algorithm IV (the influence of Algorithm II is small).

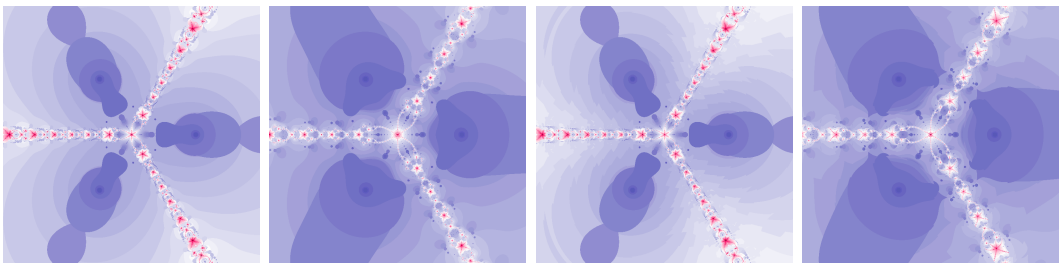


(a) $\omega_1 = 0.2$, $t = 0.50s$, (I) **(b)** $\omega_1 = 0.7$, $t = 0.48s$, (I) **(c)** $\omega_1 = 0.2$, $t = 0.57s$, (II) **(d)** $\omega_1 = 0.7$, $t = 0.54s$, (II)

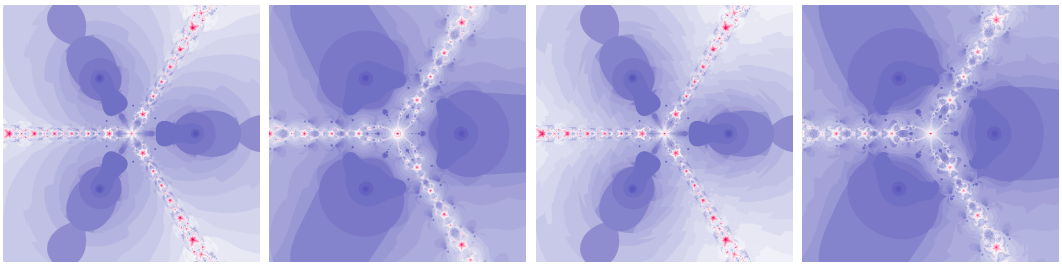


(e) $\omega_1 = 0.2$, $t = 0.54s$, (III) **(f)** $\omega_1 = 0.7$, $t = 0.53s$, (III) **(g)** $\omega_1 = 0.2$, $t = 0.60s$, (IV) **(h)** $\omega_1 = 0.7$, $t = 0.56s$, (IV)

Figure 20. Polynomiographs of the Das–Debata iteration for $\alpha = 0.6$, $\beta = 0.7$, $\eta_1 = 0.4$, $\omega_2 = 0.5$, $\eta_2 = 0.5$ and varying ω_1.



(a) $\eta_1 = 0.2$, $t = 0.69s$, (I) **(b)** $\eta_1 = 0.9$, $t = 0.49s$, (I) **(c)** $\eta_1 = 0.2$, $t = 0.79s$, (II) **(d)** $\eta_1 = 0.9$, $t = 0.54s$, (II)



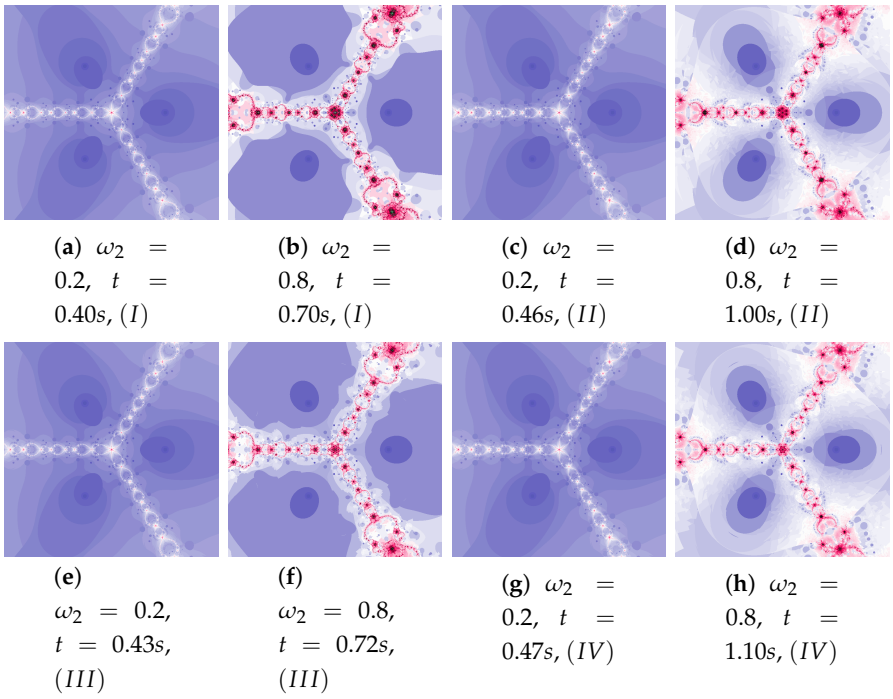
(e) $\eta_1 = 0.2$, $t = 0.75s$, (III) **(f)** $\eta_1 = 0.9$, $t = 0,52s$, (III) **(g)** $\eta_1 = 0.2$, $t = 0.85s$, (IV) **(h)** $\eta_1 = 0.9$, $t = 0.57s$, (IV)

Figure 21. Polynomiographs of the Das–Debata iteration for $\alpha = 0.5$, $\beta = 0.6$, $\omega_1 = 0.3$, $\omega_2 = 0.6$, $\eta_2 = 0.4$ and varying η_1.

The parameters ω_2 and η_2 influence the processing of the reference point. In Figure 22 polynomioographs for varying ω_2 and in Figure 23 for varying η_2 are shown. For small values of ω_2, polynomiographs presented in Figure 22a,c,e,g are similar—it means, that the operation of all the

algorithms is similar. Significant increase in the value of ω_2 shows a significant influence of the reference point on the polynomiograph's creation—Algorithms II and IV similarly create polynomiographs.

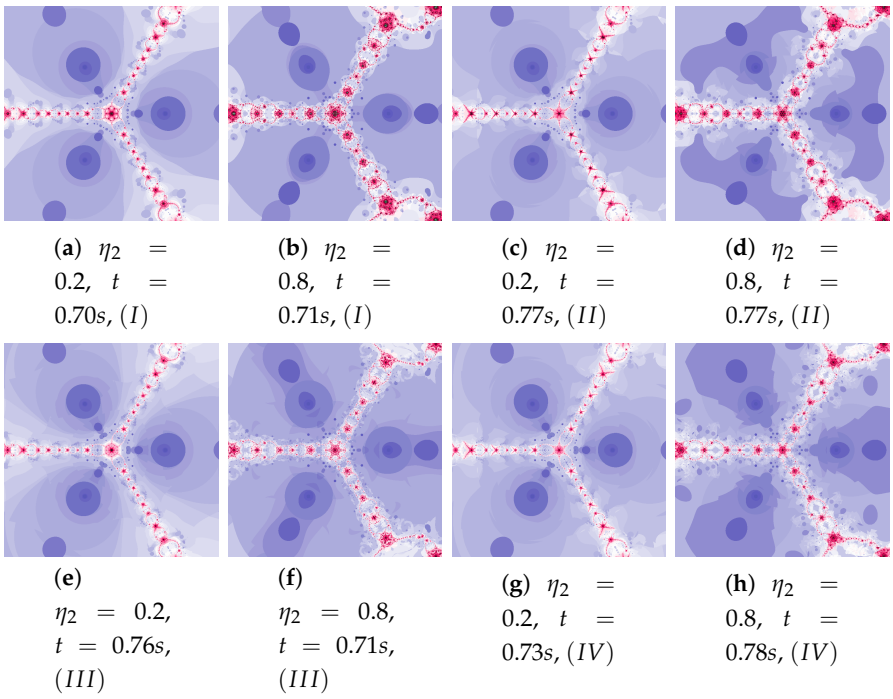
All the algorithms make characteristic changes in polynomiographs shown in Figure 23. For high value of η_2 the polynomiographs created by Algorithms II and IV are similar (see Figure 23d,).



(a) $\omega_2 = 0.2$, $t = 0.40s$, (I) **(b)** $\omega_2 = 0.8$, $t = 0.70s$, (I) **(c)** $\omega_2 = 0.2$, $t = 0.46s$, (II) **(d)** $\omega_2 = 0.8$, $t = 1.00s$, (II)

(e) $\omega_2 = 0.2$, $t = 0.43s$, (III) **(f)** $\omega_2 = 0.8$, $t = 0.72s$, (III) **(g)** $\omega_2 = 0.2$, $t = 0.47s$, (IV) **(h)** $\omega_2 = 0.8$, $t = 1.10s$, (IV)

Figure 22. Polynomiographs of the Das–Debata iteration for $\alpha = 0.9$, $\beta = 0.7$, $\omega_1 = 0.6$, $\eta_1 = 0.2$, $\eta_2 = 0.5$ and varying ω_2.

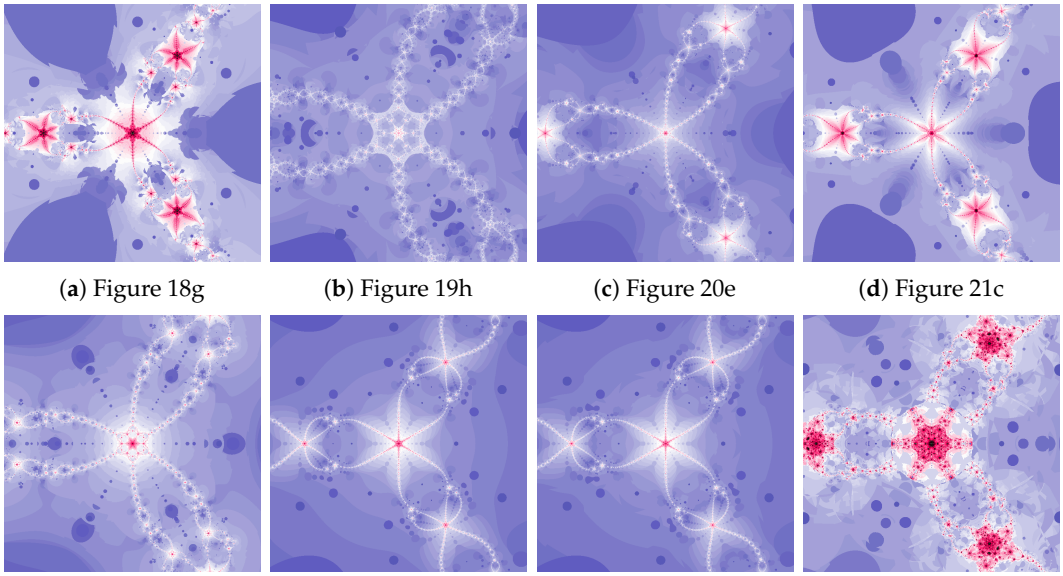


(a) $\eta_2 = 0.2$, $t = 0.70s$, (I) **(b)** $\eta_2 = 0.8$, $t = 0.71s$, (I) **(c)** $\eta_2 = 0.2$, $t = 0.77s$, (II) **(d)** $\eta_2 = 0.8$, $t = 0.77s$, (II)

(e) $\eta_2 = 0.2$, $t = 0.76s$, (III) **(f)** $\eta_2 = 0.8$, $t = 0.71s$, (III) **(g)** $\eta_2 = 0.2$, $t = 0.73s$, (IV) **(h)** $\eta_2 = 0.8$, $t = 0.78s$, (IV)

Figure 23. Polynomiographs of the Das–Debata iteration for $\alpha = 0.9$, $\beta = 0.7$, $\omega_1 = 0.4$, $\eta_1 = 0.2$, $\omega_2 = 0.5$ and varying η_2.

The magnifications of the central part of selected polynomiographs obtained with the Ishikawa and the Das–Debata iterations are presented in Figure 24. Similar to the previous examples these images have artistic features and can be used for instance in ornamentation.



(**a**) Figure 18g (**b**) Figure 19h (**c**) Figure 20e (**d**) Figure 21c

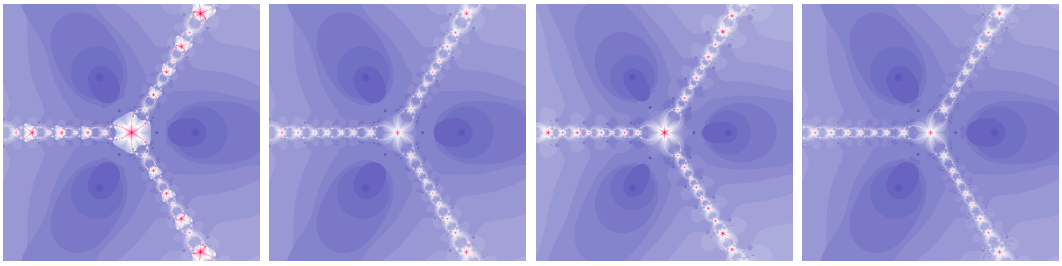
(**e**) Figure 21f (**f**) Figure 22b (**g**) Figure 22f (**h**) Figure 23h

Figure 24. Magnification of the central part of selected polynomiographs of Ishikawa and Das–Debata iterations.

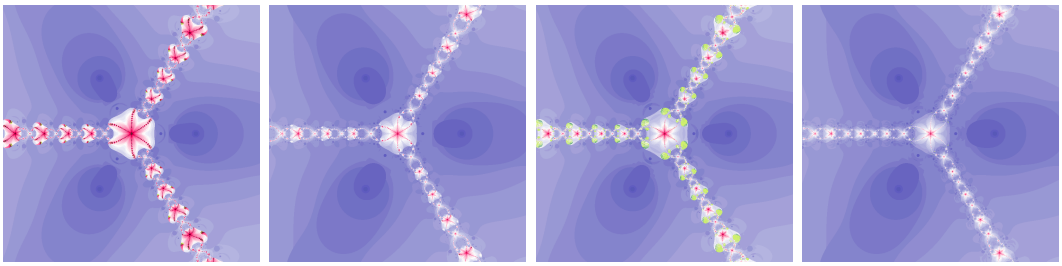
5.4. The Agarwal and the Khan–Cho–Abbas Iterations

The generalised Agarwal iteration introduces three operators parameters of which can be independently specified. It allows for a wide range of changes in particle's dynamics. In addition, α and β parameters allow to determine the impact of these operators on the algorithm work. In Figure 25 polynomiographs of the Agarwal iteration for varying α and in Figure 26 for varying β are presented. Incorrect selection of these parameters can cause that the algorithm does not reach a solution in a given number of iterations—it results in extension of the algorithm's operation time (see Figure 25e). Nevertheless, the increased particle dynamics gives the opportunity to obtain interesting patterns. Poorly selected particle dynamics can cause a significant increase in the creation time of a polynomiograph. This case is visible in the polynomiographs in Figures 27 and 28 for Algorithms III and IV.

Depending on the choice of parameters, cases that have already been discussed in the previous iterations can be observed. The big similarity of polynomiographs created by Algorithms I and II as well as III and IV is observed in Figures 29 and 30 ($\omega_2 = 0.1$). Polynomiographs are similar for all algorithms with the low particle dynamics—this is observed in Figure 31. The strong similarity of polynomiographs is shown in Figures 32 and 33 for Algorithms II and IV—it indicates a strong influence of the reference point on the polynomiographs creation. Next, the strong similarity of polynomiographs is observed in Figure 34 for Algorithm I and III—it results from the strong influence of the best particle position on the polynomiographs creation.

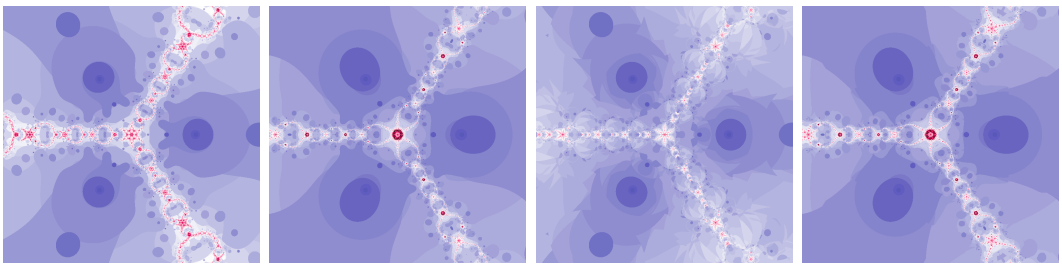


(a) $\beta = 0.3$, $t = 0.60s$, (I) **(b)** $\beta = 0.7$, $t = 0.55s$, (I) **(c)** $\beta = 0.3$, $t = 0.66s$, (II) **(d)** $\beta = 0.7$, $t = 0.59s$, (II)

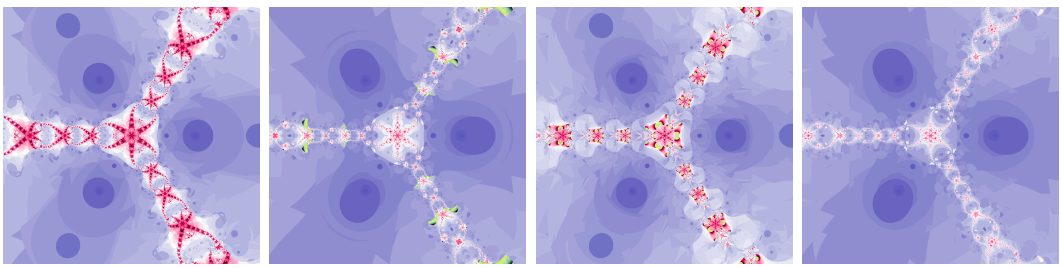


(e) $\beta = 0.3$, $t = 0.74s$, (III) **(f)** $\beta = 0.7$, $t = 0.62s$, (III) **(g)** $\beta = 0.3$, $t = 0.93s$, (IV) **(h)** $\beta = 0.7$, $t = 0.61s$, (IV)

Figure 25. Polynomiographs of the Agarwal iteration for $\beta = 0.5$, $\omega_1 = 0.4$, $\eta_1 = 0.3$, $\omega_2 = 0.4$, $\eta_2 = 0.3$, $\omega_3 = 0.4$, $\eta_3 = 0.3$ and varying α.

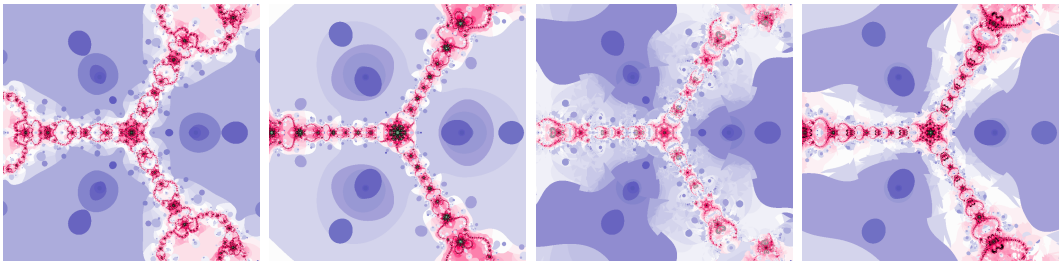


(a) $\beta = 0.1$, $t = 0.79s$, (I) **(b)** $\beta = 0.9$, $t = 0.64s$, (I) **(c)** $\beta = 0.1$, $t = 0.83s$, (II) **(d)** $\beta = 0.9$, $t = 0.71s$, (II)



(e) $\beta = 0.1$, $t = 0.99s$, (III) **(f)** $\beta = 0.9$, $t = 0.81s$, (III) **(g)** $\beta = 0.1$, $t = 1.09s$, (IV) **(h)** $\beta = 0.9$, $t = 0.78s$, (IV)

Figure 26. Polynomiographs of the Agarwal iteration for $\alpha = 0.5$, $\omega_1 = 0.5$, $\eta_1 = 0.5$, $\omega_2 = 0.5$, $\eta_2 = 0.5$, $\omega_3 = 0.5$, $\eta_3 = 0.5$ and varying β.

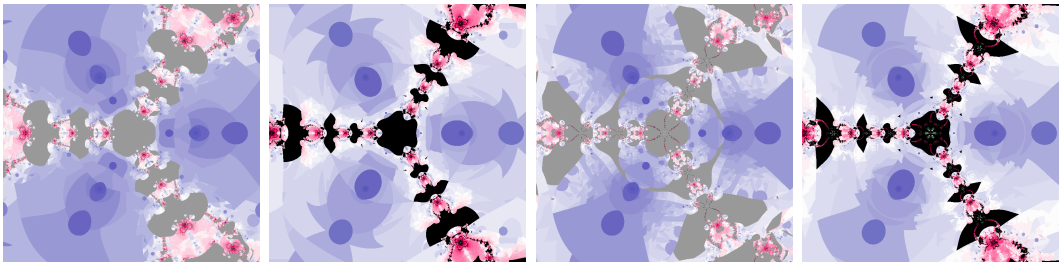


(a) $\beta = 0.8$, $\omega_1 = 0.7$, $\eta_1 = 0.2$, $\eta_2 = 0.7$, $\eta_3 = 0.5$, $t = 1.10s$, (I)

(b) $\beta = 0.5$, $\omega_1 = 0.8$, $\eta_1 = 0.4$, $\eta_2 = 0.3$, $\eta_3 = 0.3$, $t = 1.21s$, (I)

(c) $\beta = 0.8$, $\omega_1 = 0.7$, $\eta_1 = 0.2$, $\eta_2 = 0.7$, $\eta_3 = 0.5$, $t = 1.24s$, (II)

(d) $\beta = 0.5$, $\omega_1 = 0.8$, $\eta_1 = 0.4$, $\eta_2 = 0.3$, $\eta_3 = 0.3$, $t = 1.28s$, (II)



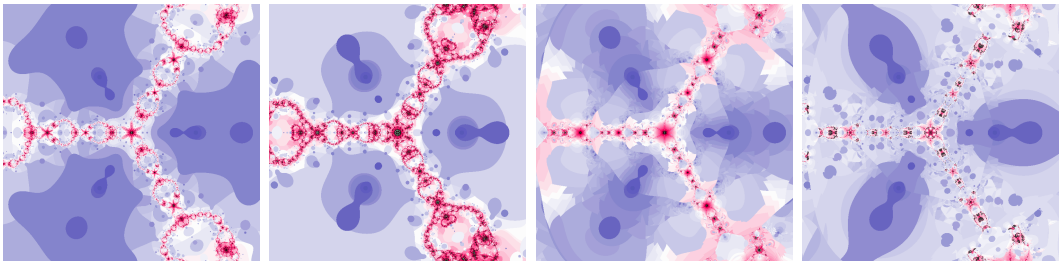
(e) $\beta = 0.8$, $\omega_1 = 0.7$, $\eta_1 = 0.2$, $\eta_2 = 0.7$, $\eta_3 = 0.5$, $t = 4.74s$, (III)

(f) $\beta = 0.5$, $\omega_1 = 0.8$, $\eta_1 = 0.4$, $\eta_2 = 0.3$, $\eta_3 = 0.3$, $t = 3.41s$, (III)

(g) $\beta = 0.8$, $\omega_1 = 0.7$, $\eta_1 = 0.2$, $\eta_2 = 0.7$, $\eta_3 = 0.5$, $t = 7.05s$, (IV)

(h) $\beta = 0.5$, $\omega_1 = 0.8$, $\eta_1 = 0.4$, $\eta_2 = 0.3$, $\eta_3 = 0.3$, $t = 3.43s$, (IV)

Figure 27. Polynomiographs of the generalised Agarwal iteration for $\alpha = 0.5, \omega_2 = 0.6, \eta_2 = 0.7, \omega_3 = 0.9$.

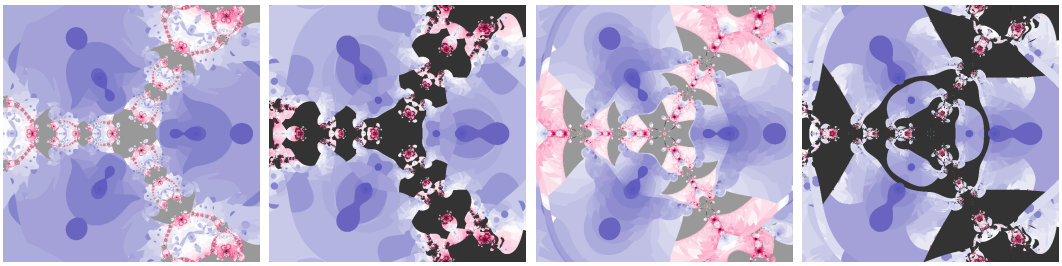


(a) $\alpha = 0.8$, $\beta = 0.2$, $t = 0.87s$, (I)

(b) $\alpha = 0.5$, $\beta = 0.3$, $t = 1.28s$, (I)

(c) $\alpha = 0.8$, $\beta = 0.2$, $t = 1.20s$, (II)

(d) $\alpha = 0.5$, $\beta = 0.3$, $t = 1.29s$, (II)



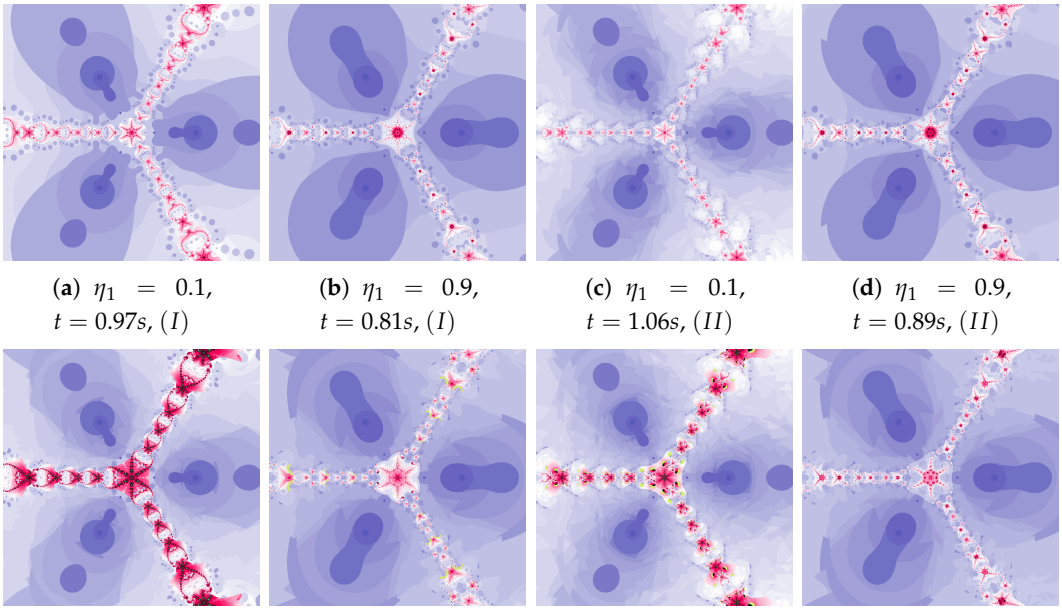
(e) $\alpha = 0.8$, $\beta = 0.2$, $t = 2.95s$, (III)

(f) $\alpha = 0.5$, $\beta = 0.3$, $t = 5.96s$, (III)

(g) $\alpha = 0.8$, $\beta = 0.2$, $t = 4.50s$, (IV)

(h) $\alpha = 0.5$, $\beta = 0.3$, $t = 8.82s$, (IV)

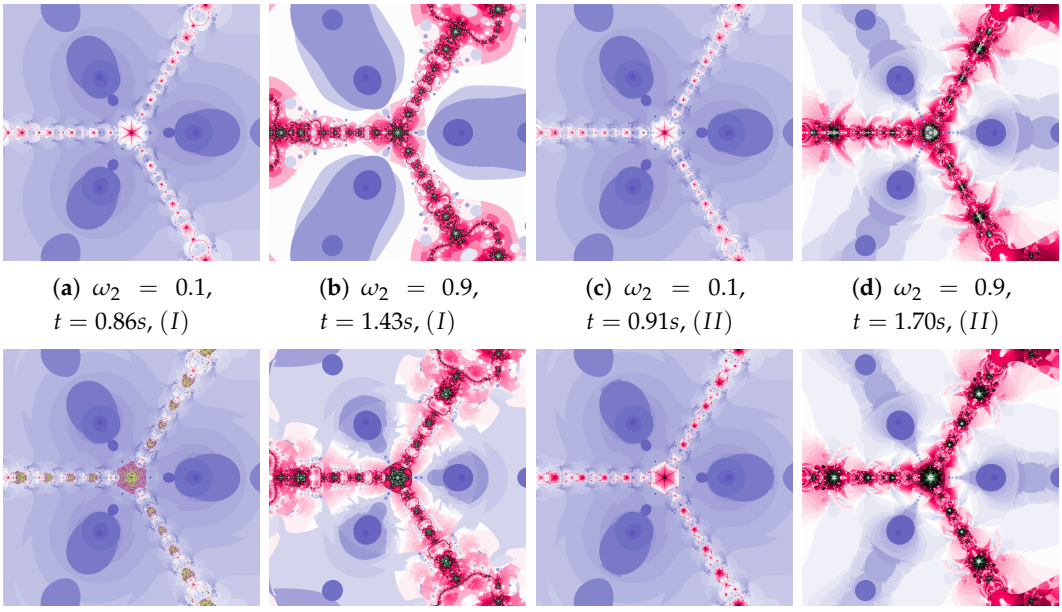
Figure 28. Polynomiographs of the generalised Agarwal iteration for $\omega_1 = 0.3, \eta_1 = 0.4, \omega_2 = 0.6, \eta_2 = 0.7, \omega_3 = 0.9, \eta_3 = 0.5$.



(a) $\eta_1 = 0.1$, $t = 0.97s$, (I) **(b)** $\eta_1 = 0.9$, $t = 0.81s$, (I) **(c)** $\eta_1 = 0.1$, $t = 1.06s$, (II) **(d)** $\eta_1 = 0.9$, $t = 0.89s$, (II)

(e) $\eta_1 = 0.1$, $t = 1.21s$, (III) **(f)** $\eta_1 = 0.9$, $t = 0.96s$, (III) **(g)** $\eta_1 = 0.1$, $t = 1.33s$, (IV) **(h)** $\eta_1 = 0.9$, $t = 0.98s$, (IV)

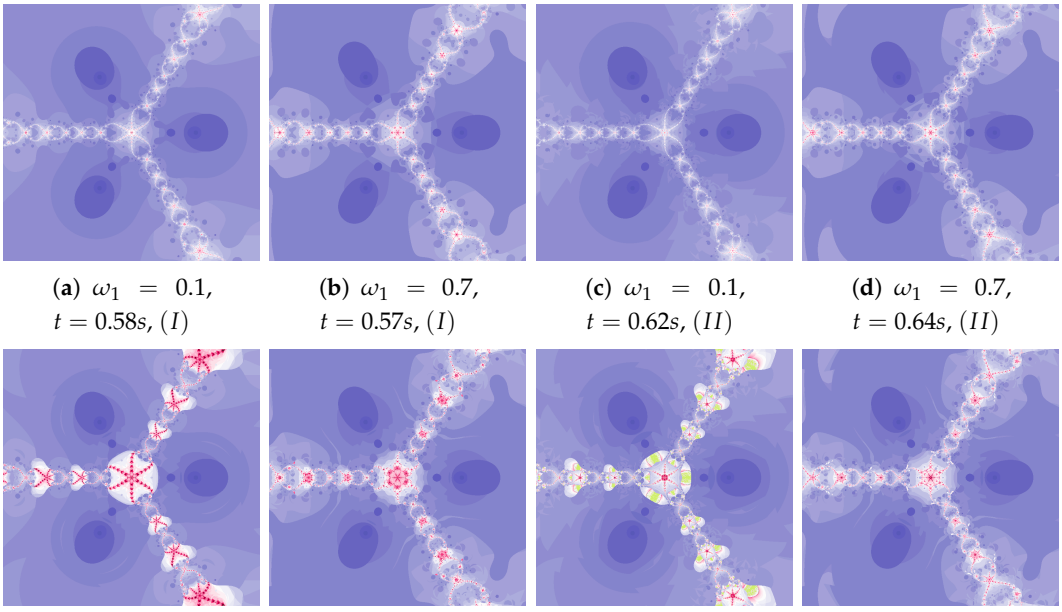
Figure 29. Polynomiographs of the generalised Agarwal iteration for $\alpha = 0.5$, $\beta = 0.5$, $\omega_1 = 0.6$, $\omega_2 = 0.6$, $\eta_2 = 0.3$, $\omega_3 = 0.6$, $\eta_3 = 0.3$ ($T_2 = T_3$) and varying η_1.



(a) $\omega_2 = 0.1$, $t = 0.86s$, (I) **(b)** $\omega_2 = 0.9$, $t = 1.43s$, (I) **(c)** $\omega_2 = 0.1$, $t = 0.91s$, (II) **(d)** $\omega_2 = 0.9$, $t = 1.70s$, (II)

(e) $\omega_2 = 0.1$, $t = 1.25s$, (III) **(f)** $\omega_2 = 0.9$, $t = 1.61s$, (III) **(g)** $\omega_2 = 0.1$, $t = 1.03s$, (IV) **(h)** $\omega_2 = 0.9$, $t = 2.02s$, (IV)

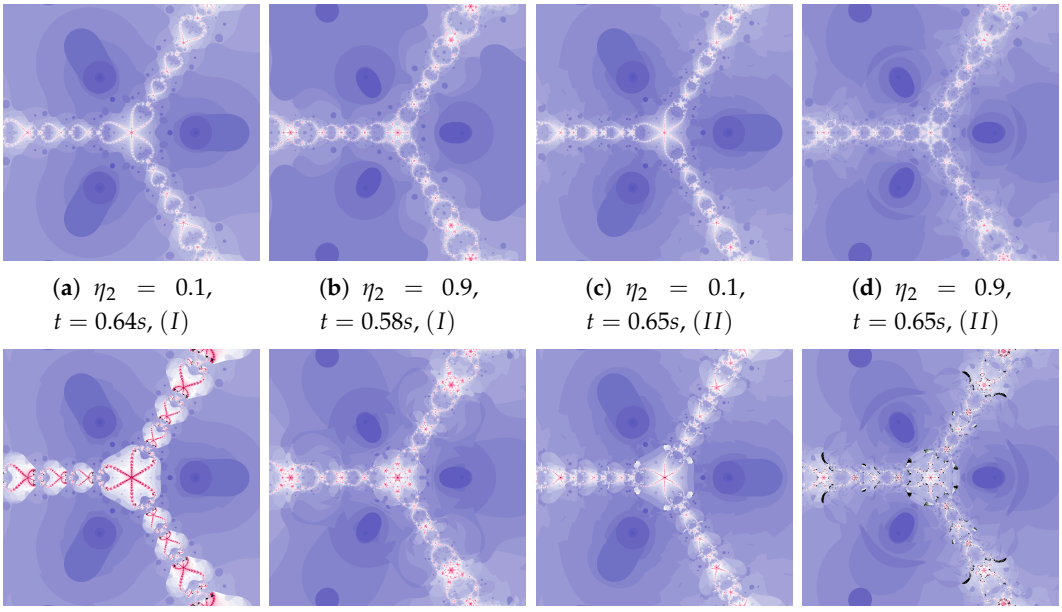
Figure 30. Polynomiographs of the Khan–Cho–Abbas iteration for $\alpha = 0.5$, $\beta = 0.5$, $\omega_1 = 0.7$, $\eta_1 = 0.2$, $\eta_2 = 0.5$, $\omega_3 = 0.7$, $\eta_3 = 0.2$ and varying ω_2.



(a) $\omega_1 = 0.1$, $t = 0.58s$, (I) **(b)** $\omega_1 = 0.7$, $t = 0.57s$, (I) **(c)** $\omega_1 = 0.1$, $t = 0.62s$, (II) **(d)** $\omega_1 = 0.7$, $t = 0.64s$, (II)

(e) $\omega_1 = 0.1$, $t = 0.73s$, (III) **(f)** $\omega_1 = 0.7$, $t = 0.66s$, (III) **(g)** $\omega_1 = 0.1$, $t = 0.96s$, (IV) **(h)** $\omega_1 = 0.7$, $t = 0.71s$, (IV)

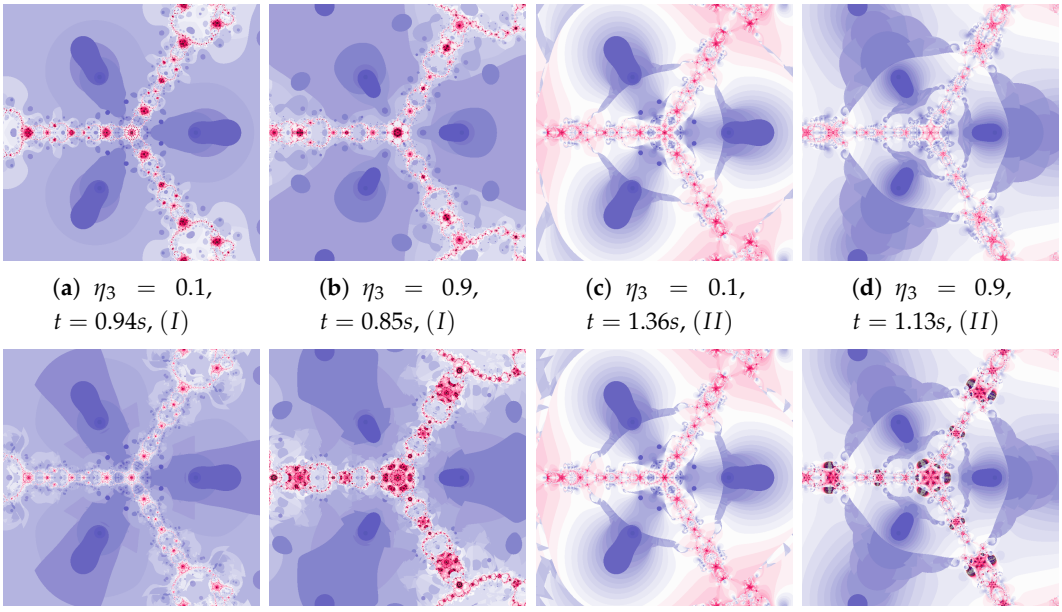
Figure 31. Polynomiographs of the generalised Agarwal iteration for $\alpha = 0.5$, $\beta = 0.5$, $\eta_1 = 0.6$, $\omega_2 = 0.4$, $\eta_2 = 0.6$, $\omega_3 = 0.4$, $\eta_3 = 0.6$ ($T_2 = T_3$) and varying ω_1.



(a) $\eta_2 = 0.1$, $t = 0.64s$, (I) **(b)** $\eta_2 = 0.9$, $t = 0.58s$, (I) **(c)** $\eta_2 = 0.1$, $t = 0.65s$, (II) **(d)** $\eta_2 = 0.9$, $t = 0.65s$, (II)

(e) $\eta_2 = 0.1$, $t = 0.80s$, (III) **(f)** $\eta_2 = 0.9$, $t = 0.69s$, (III) **(g)** $\eta_2 = 0.1$, $t = 0.75s$, (IV) **(h)** $\eta_2 = 0.9$, $t = 0.96s$, (IV)

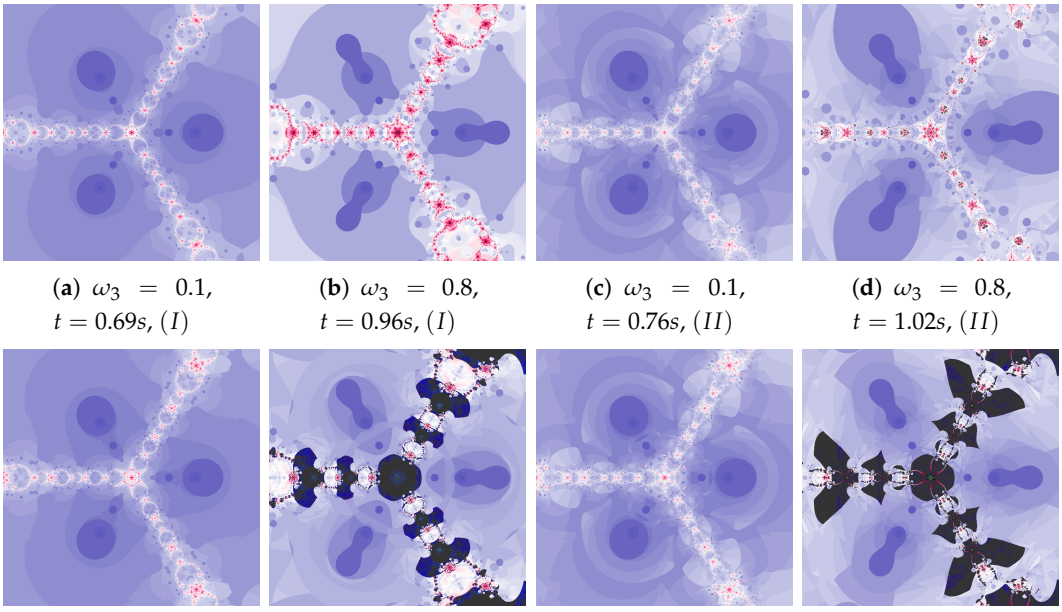
Figure 32. Polynomiographs of the Khan–Cho–Abbas iteration for $\alpha = 0.5$, $\beta = 0.5$, $\omega_1 = 0.3$, $\eta_1 = 0.7$, $\omega_2 = 0.6$, $\omega_3 = 0.3$, $\eta_3 = 0.7$ and varying η_2.



(a) $\eta_3 = 0.1$, $t = 0.94s$, (I) **(b)** $\eta_3 = 0.9$, $t = 0.85s$, (I) **(c)** $\eta_3 = 0.1$, $t = 1.36s$, (II) **(d)** $\eta_3 = 0.9$, $t = 1.13s$, (II)

(e) $\eta_3 = 0.1$, $t = 0.90s$, (III) **(f)** $\eta_3 = 0.9$, $t = 0.99s$, (III) **(g)** $\eta_3 = 0.1$, $t = 1.49s$, (IV) **(h)** $\eta_3 = 0.9$, $t = 1.50s$, (IV)

Figure 33. Polynomiographs of the generalised Agarwal iteration for $\alpha = 0.8$, $\beta = 0.4$, $\omega_1 = 0.2$, $\eta_1 = 0.4$, $\omega_2 = 0.7$, $\eta_2 = 0.7$, $\omega_3 = 0.5$ and varying η_3.

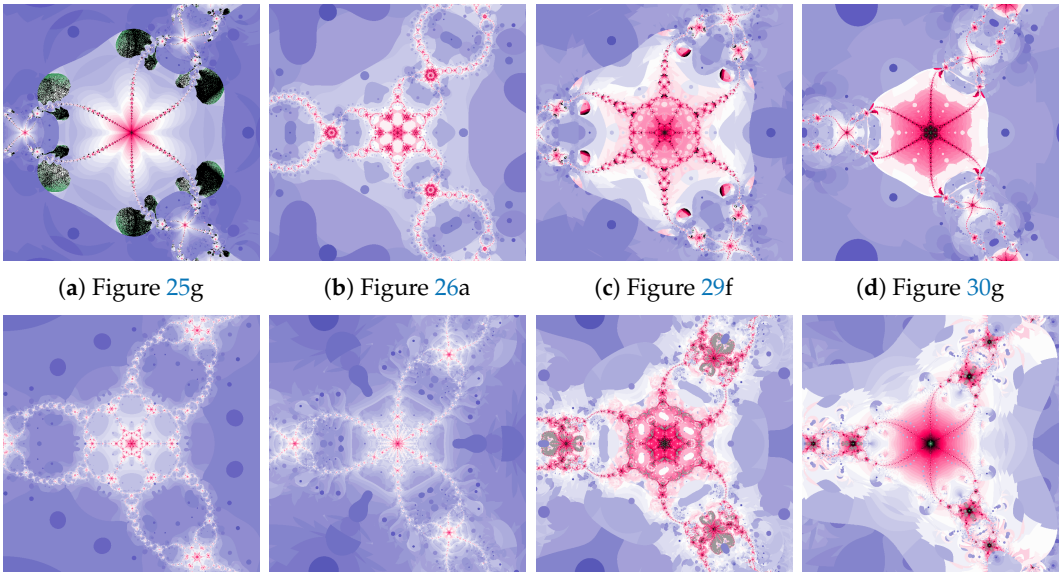


(a) $\omega_3 = 0.1$, $t = 0.69s$, (I) **(b)** $\omega_3 = 0.8$, $t = 0.96s$, (I) **(c)** $\omega_3 = 0.1$, $t = 0.76s$, (II) **(d)** $\omega_3 = 0.8$, $t = 1.02s$, (II)

(e) $\omega_3 = 0.1$, $t = 0.70s$, (III) **(f)** $\omega_3 = 0.8$, $t = 4.06s$, (III) **(g)** $\omega_3 = 0.1$, $t = 0.91s$, (IV) **(h)** $\omega_3 = 0.8$, $t = 5.63s$, (IV)

Figure 34. Polynomiographs of the generalised Agarwal iteration for $\alpha = 0.5$, $\beta = 0.3$, $\omega_1 = 0.4$, $\eta_1 = 0.6$, $\omega_2 = 0.6$, $\eta_2 = 0.7$, $\eta_3 = 0.5$ and varying ω_3.

The magnifications of the central part of selected polynomiographs of Agarwal and Khan–Cho–Abbas iterations are presented in Figure 35. High dynamics can create interesting patterns, especially for the Agarwal iteration. These patterns can be an artistic inspiration.



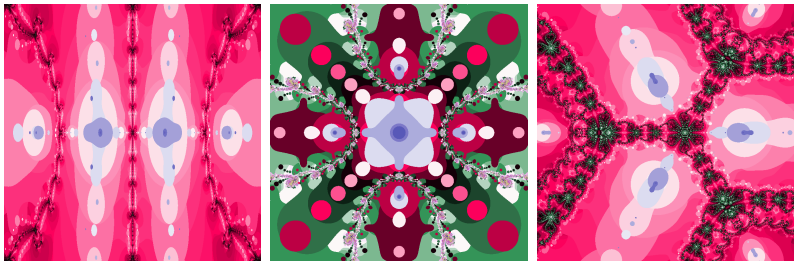
(**a**) Figure 25g (**b**) Figure 26a (**c**) Figure 29f (**d**) Figure 30g

(**e**) Figure 32b (**f**) Figure 34c (**g**) Figure 27d (**h**) Figure 28c

Figure 35. Magnification of the central part of selected polynomiographs of Agarwal and Khan–Cho–Abbas iterations.

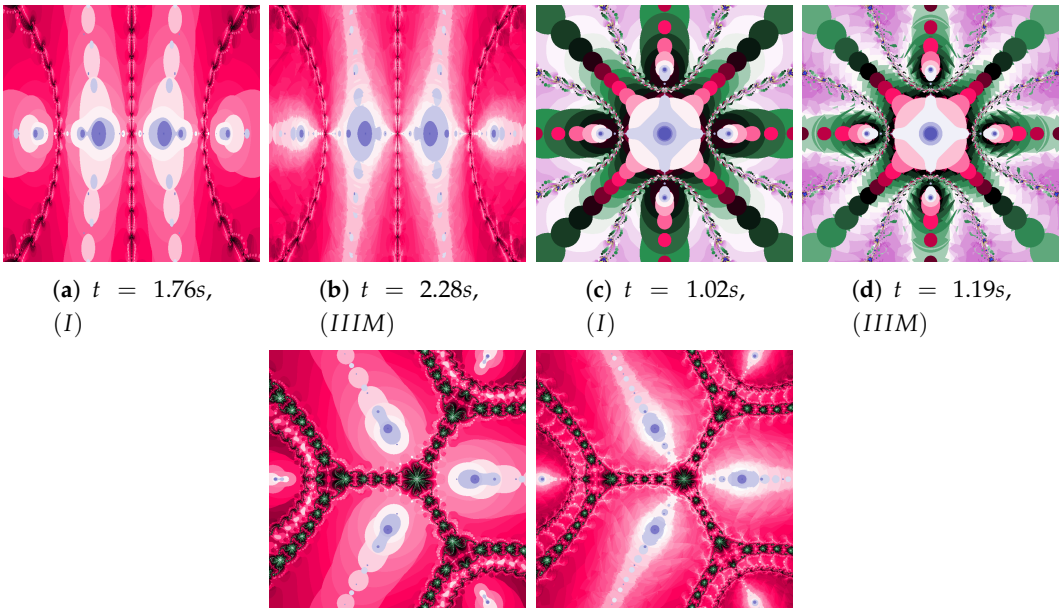
5.5. Algorithms Operation in the Selected Test Environments

The discussion conducted in the previous sections presented in detail the impact of individual factors on the operation of the proposed algorithm. The authors want to supplement this discussion with a presentation of algorithms' operation for the selected test environments, which we can be found in the world literature and were presented at the beginning of Section 5. Figures 36–42 present polynomiographs for three test environments and the iterations discussed in the article. They will allow to generalise the observations discussed in the previous sections. In Figure 36 the Picard's iteration, in Figure 37a,c,e the Mann's iteration and in sub-figures (a), (e), (i) of Figures 38–42 polynomiographs realised by the base algorithm are presented. These polynomiographs are characterised by smooth boundaries of areas with different particle dynamics. The implementation of the algorithm's modification consisting in the selection of position of the best reference point and particle causes the blur and rag of smooth edges of the areas presented on the polynomiographs obtained for the base algorithm. The selection of the best position of the reference point implemented in the Algorithm (II) results in a significant reduction of small elements (details) of the patterns visible on the polynomiograph. It is a significant limitation of particle dynamics, which can be seen in the (b) sub-figures of Figures 38–42. Figures 37b,d,f and the (c) sub-figures of Figures 38–42 present polynomiographs of the Algorithm ($IIIM$ and III) selecting the best particle position. The operation of the algorithm results in much better presentation of small details in the polynomiograph—these are the areas in which the particle wants to achieve a worse position. Polynomiographs obtained by using the algorithm denoted by (IV), i.e., with the selection of the best reference point and the best particle position, are shown in the (d) sub-figures of Figures 38–42. We can observe both features of the polynomiographs from the sub-figures (b) and (c) of Figures 38–42. However, the polynomiograph's features have a dominant influence for choosing the best particle position. The analysis of the changes in particle dynamics resulting from the applied algorithm can also be performed by analysing the simulation time. The best position of the reference point has a significant impact on the reduction of particle dynamics. One can only complete the observations regarding the test environments. In environments $p(c) = c^3 - 1$, $p(c) = c^4 - 10c^2 + 9$, $p(c) = c^6 + 10c^3 - 8$ the particle obtains similar ranges of dynamics, whereas for the $p(c) = c^5 - c$ environment the particle achieves larger dynamics ranges.



(a) $t = 1.67s, (I)$ **(b)** $t = 0.82s$, (II) **(c)** $t = 3.44s$, (III)

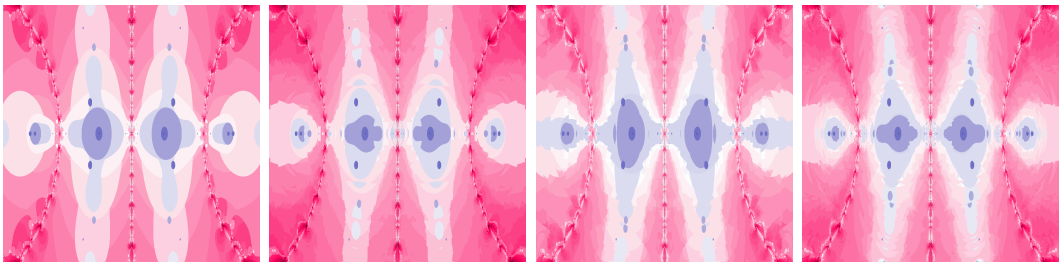
Figure 36. Polynomiographs of the Picard iteration for the polynomiograph settings from the Figure 4d.



(a) $t = 1.76s$, (I) **(b)** $t = 2.28s$, $(IIIM)$ **(c)** $t = 1.02s$, (I) **(d)** $t = 1.19s$, $(IIIM)$

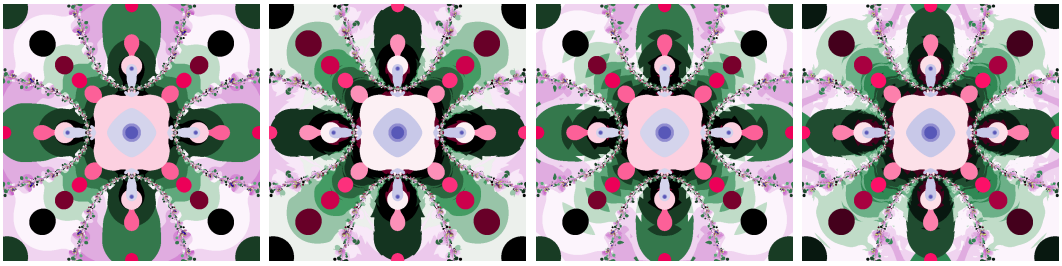
(e) $t = 3.48s$, (I) **(f)** $t = 6.02s$, $(IIIM)$

Figure 37. Polynomiographs of the Mann iteration for the polynomiograph settings from the Figure 15b.

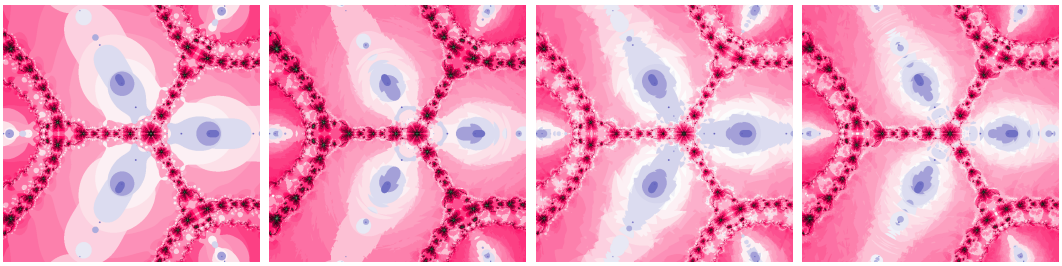


(a) $t = 2.39s$, (I) **(b)** $t = 2.83s$, (II) **(c)** $t = 2.61s$, (III) **(d)** $t = 2.96s$, (IV)

Figure 38. *Cont.*

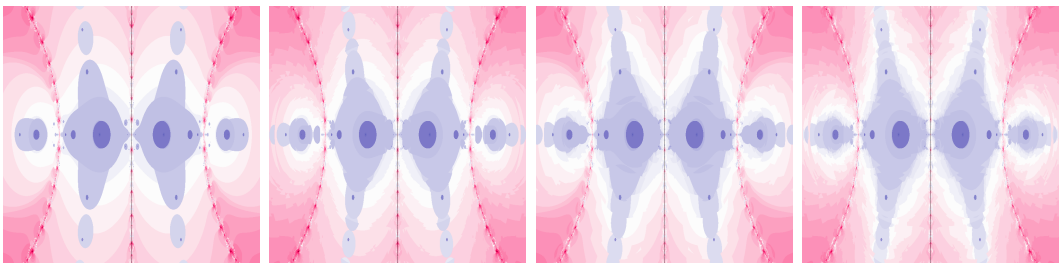


(e) $t = 1.29s$, (I) **(f)** $t = 1.49s$, (II) **(g)** $t = 1.43s$, (III) **(h)** $t = 1.59s$, (IV)

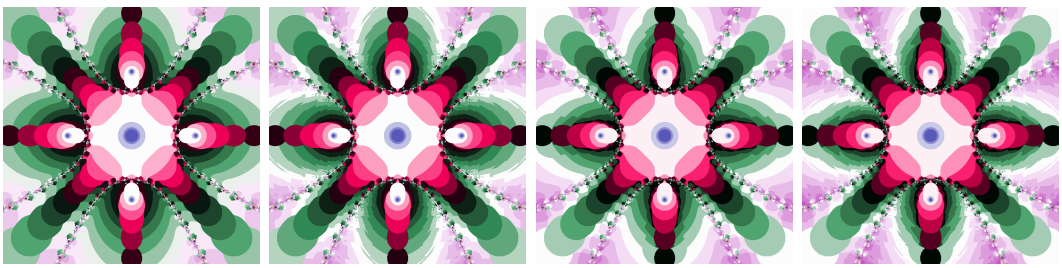


(i) $t = 4.79s$, (I) **(j)** $t = 5.60s$, (II) **(k)** $t = 5.00s$, (III) **(l)** $t = 5.81s$, (IV)

Figure 38. Polynomiographs of the Ishikawa iteration for the polynomiograph settings from the Figure 18b.

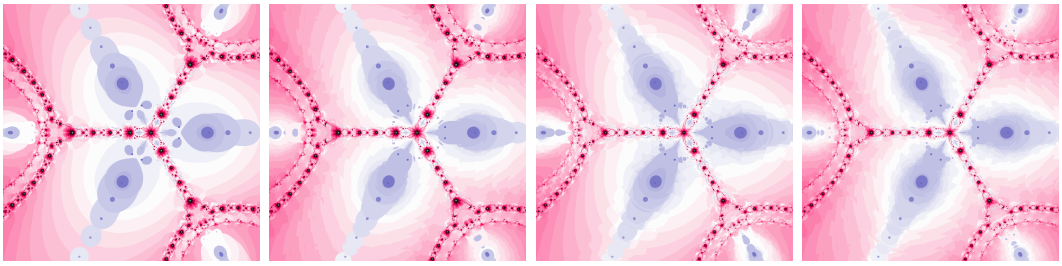


(a) $t = 1.98s$, (I) **(b)** $t = 2.24s$, (II) **(c)** $t = 2.16s$, (III) **(d)** $t = 2.43s$, (IV)



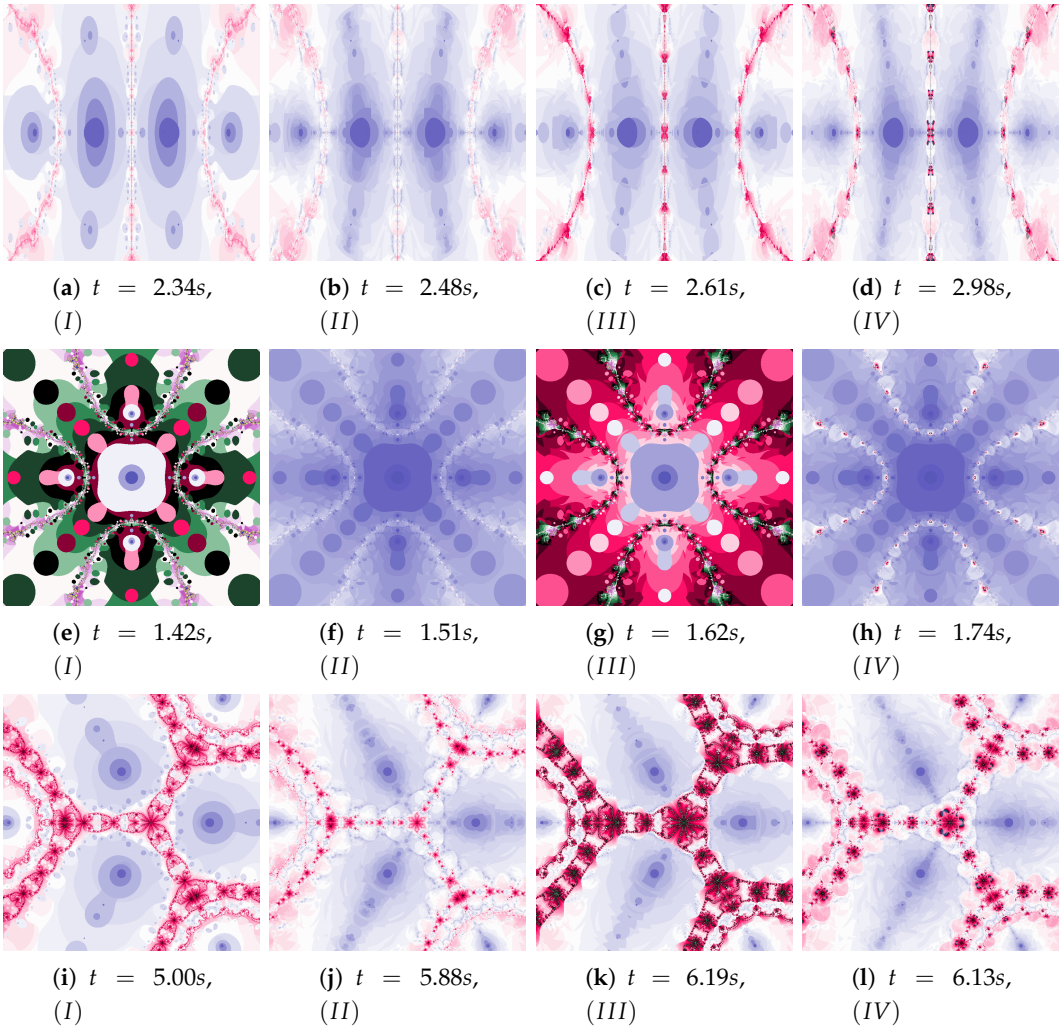
(e) $t = 1.38s$, (I) **(f)** $t = 1.53s$, (II) **(g)** $t = 1.49s$, (III) **(h)** $t = 1.67s$, (IV)

Figure 39. *Cont.*



(i) $t = 3.89s$, (I) **(j)** $t = 4.34s$, (II) **(k)** $t = 4.17s$, (III) **(l)** $t = 4.63s$, (IV)

Figure 39. Polynomiographs of the Das–Debata iteration for the polynomiograph settings from the Figure 21a.

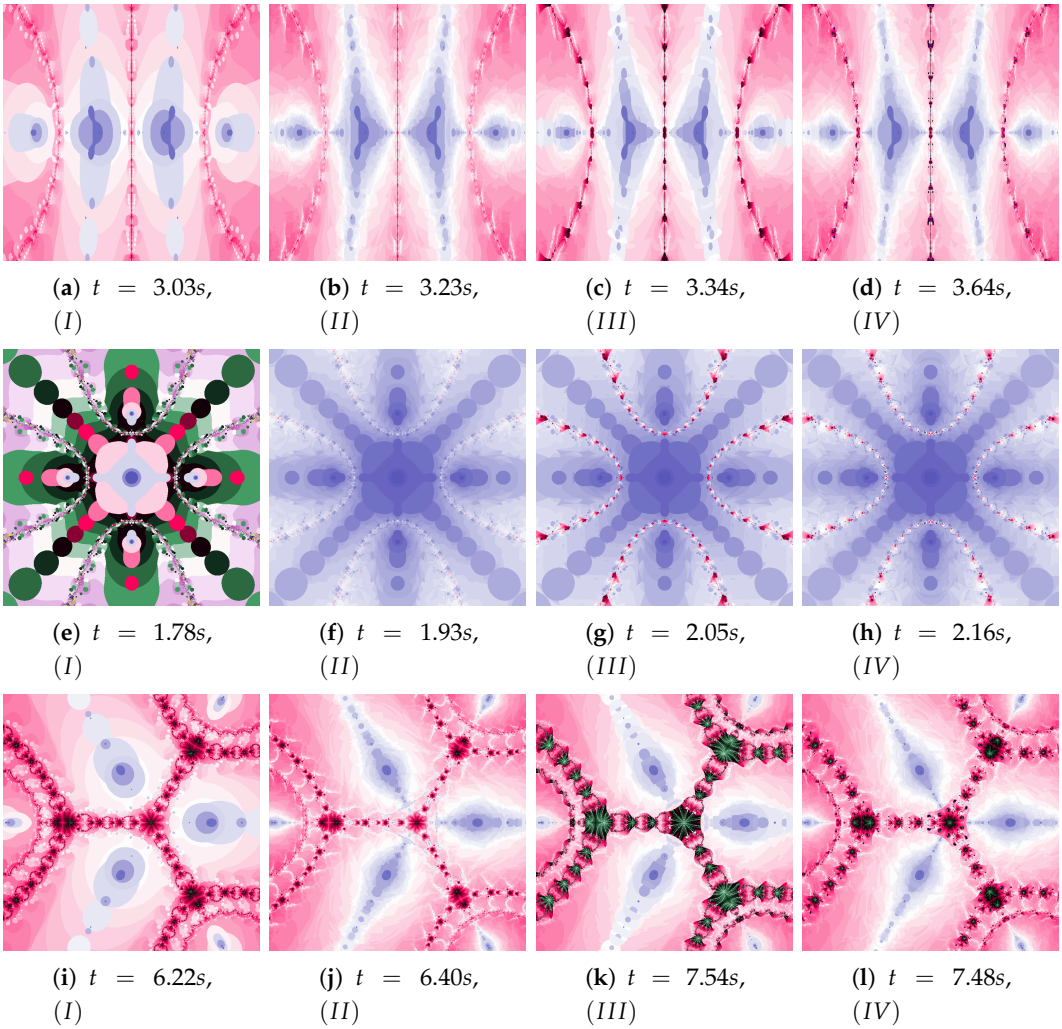


(a) $t = 2.34s$, (I) **(b)** $t = 2.48s$, (II) **(c)** $t = 2.61s$, (III) **(d)** $t = 2.98s$, (IV)

(e) $t = 1.42s$, (I) **(f)** $t = 1.51s$, (II) **(g)** $t = 1.62s$, (III) **(h)** $t = 1.74s$, (IV)

(i) $t = 5.00s$, (I) **(j)** $t = 5.88s$, (II) **(k)** $t = 6.19s$, (III) **(l)** $t = 6.13s$, (IV)

Figure 40. Polynomiographs of the Agarwal iteration for the polynomiograph settings from the Figure 26a.

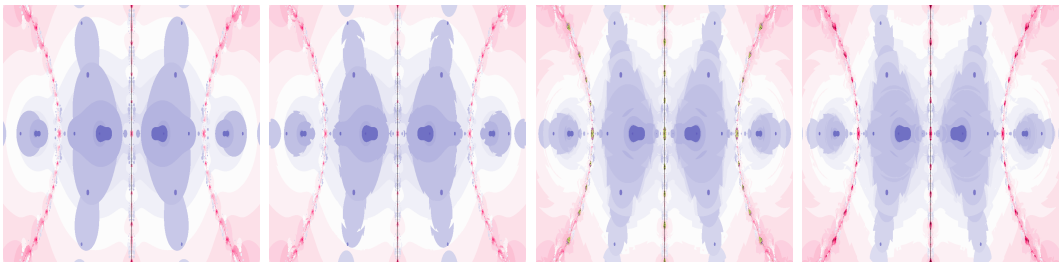


(a) $t = 3.03s$, (I) **(b)** $t = 3.23s$, (II) **(c)** $t = 3.34s$, (III) **(d)** $t = 3.64s$, (IV)

(e) $t = 1.78s$, (I) **(f)** $t = 1.93s$, (II) **(g)** $t = 2.05s$, (III) **(h)** $t = 2.16s$, (IV)

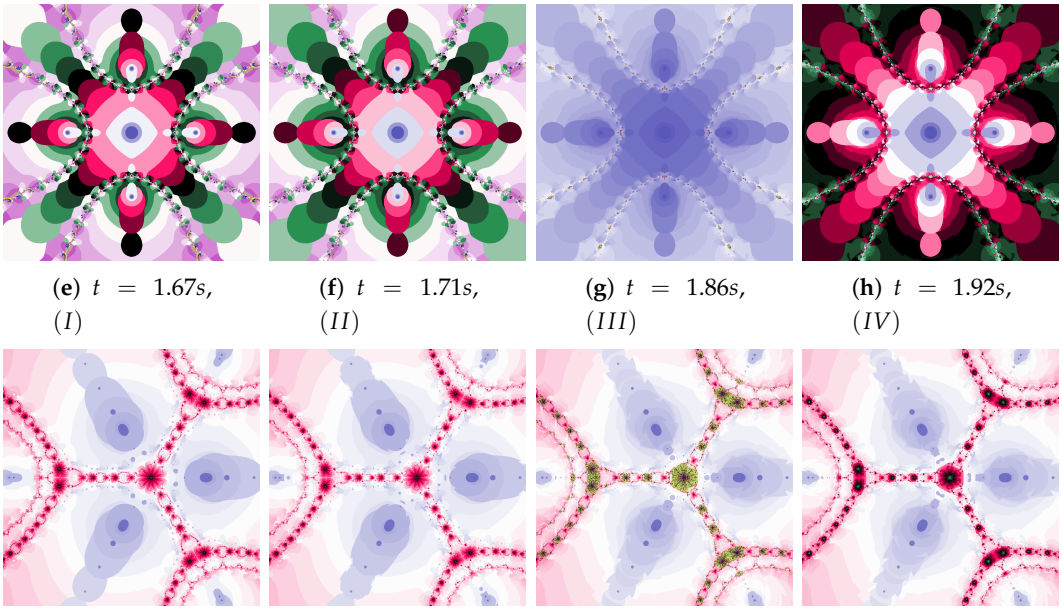
(i) $t = 6.22s$, (I) **(j)** $t = 6.40s$, (II) **(k)** $t = 7.54s$, (III) **(l)** $t = 7.48s$, (IV)

Figure 41. Polynomiographs of the generalised Agarwal iteration for the polynomiograph settings from the Figure 29a.



(a) $t = 2.50s$, (I) **(b)** $t = 2.67s$, (II) **(c)** $t = 2.86s$, (III) **(d)** $t = 2.94s$, (IV)

Figure 42. *Cont.*



(e) $t = 1.67s$, (I) **(f)** $t = 1.71s$, (II) **(g)** $t = 1.86s$, (III) **(h)** $t = 1.92s$, (IV)

(i) $t = 5.04s$, (I) **(j)** $t = 5.32s$, (II) **(k)** $t = 7.35s$, (III) **(l)** $t = 6.16s$, (IV)

Figure 42. Polynomiographs of the Khan–Cho–Abbas iteration for the polynomiograph settings from the Figure 30a.

6. Conclusions

The paper proposes a modification of the Newton's method and introduces algorithms based on it. Similar approaches are presented in many optimisation algorithms which require parameters tuning, e.g., the PSO algorithm. The discussion presented in the article allows showing inertia weight and the acceleration constant impact on the operation of the proposed algorithms. Both too large and too small dynamics of particle can have an adverse influence on the algorithm's operation. Polynomiography is a tool illustrating the behaviour of particles. It also creates an artistic mosaics and allows obtaining images that can be named as art.

Author Contributions: All the authors have equally contributed. All authors have read and agree to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Polak, E. *Optimization Algorithms and Consistent Approximations*; Springer-Verlag: New York, NY, USA, 1997, doi:10.1007/978-1-4612-0663-7.
2. Gosciniak, I. Discussion on semi-immune algorithm behaviour based on fractal analysis. *Soft Comput.* **2017**, *21*, 3945–3956, doi:10.1007/s00500-016-2044-y.
3. Weise, T. *Global Optimization Algorithms—Theory and Application*, 2nd ed.; Self-Published, 2009. Available online: http://www.it-weise.de/projects/book.pdf (accessed on 01 May 2020).
4. Zhang, W.; Ma, D.; Wei, J.; Liang, H. A Parameter Selection Strategy for Particle Swarm Optimization Based on Particle Positions. *Expert Syst. Appl.* **2014**, *41*, 3576–3584, doi:10.1016/j.eswa.2013.10.061.
5. Van den Bergh, F.; Engelbrecht, A.P. A Convergence Proof for the Particle Swarm Optimiser. *Fundam. Inf.* **2010**, *105*, 341–374.
6. Freitas, D.; Lopes, L.; Morgado-Dias, F. Particle Swarm Optimisation: A Historical Review Up to the Current Developments. *Entropy* **2020**, *22*, 362, doi:10.3390/e22030362.

7. Cheney, W.; Kincaid, D. *Numerical Mathematics and Computing*, 6th ed.; Brooks/Cole: Pacific Groove, CA, USA, 2007.
8. Goh, B.; Leong, W.; Siri, Z. Partial Newton Methods for a System of Equations. *Numer. Algebr. Control Optim.* **2013**, *3*, 463–469, doi:10.3934/naco.2013.3.463.
9. Saheya, B.; Chen, G.Q.; Sui, Y.K.; Wu, C.Y. A New Newton-like Method for Solving Nonlinear Equations. *SpringerPlus* **2016**, *5*, 1269, doi:10.1186/s40064-016-2909-7.
10. Sharma, J.; Sharma, R.; Bahl, A. An Improved Newton-Traub Composition for Solving Systems of Nonlinear Equations. *Appl. Math. Comput.* **2016**, *290*, 98–110, doi:10.1016/j.amc.2016.05.051.
11. Abbasbandy, S.; Bakhtiari, P.; Cordero, A.; Torregrosa, J.; Lotfi, T. New Efficient Methods for Solving Nonlinear Systems of Equations with Arbitrary Even Order. *Appl. Math. Comput.* **2016**, *287–288*, 94–103, doi:10.1016/j.amc.2016.04.038.
12. Xiao, X.Y.; Yin, H.W. Accelerating the Convergence Speed of Iterative Methods for Solving Nonlinear Systems. *Appl. Math. Comput.* **2018**, *333*, 8–19, doi:10.1016/j.amc.2018.03.108.
13. Alqahtani, H.; Behl, R.; Kansal, M. Higher-Order Iteration Schemes for Solving Nonlinear Systems of Equations. *Mathematics* **2019**, *7*, 937, doi:10.3390/math7100937.
14. Awwal, A.; Wang, L.; Kumam, P.; Mohammad, H. A Two-Step Spectral Gradient Projection Method for System of Nonlinear Monotone Equations and Image Deblurring Problems. *Symmetry* **2020**, *12*, 874, doi:10.3390/sym12060874.
15. Wang, Q.; Zeng, J.; Jie, J. Modified Particle Swarm Optimization for Solving Systems of Equations. In *Advanced Intelligent Computing Theories and Applications*; Volume 2: Communications in Computer and Information Science; Huang, D., Heutte, L., Loog, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 361–369, doi:10.1007/978-3-540-74282-1_41.
16. Ouyang, A.; Zhou, Y.; Luo, Q. Hybrid Particle Swarm Optimization Algorithm for Solving Systems of Nonlinear Equations. In Proceedings of the 2009 IEEE International Conference on Granular Computing, Nanchang, China, 17–19 August 2009; pp. 460–465, doi:10.1109/GRC.2009.5255079.
17. Jaberipour, M.; Khorram, E.; Karimi, B. Particle Swarm Algorithm for Solving Systems of Nonlinear Equations. *Comput. Math. Appl.* **2011**, *62*, 566–576, doi:10.1016/j.camwa.2011.05.031.
18. Li, Y.; Wei, Y.; Chu, Y. Research on Solving Systems of Nonlinear Equations Based on Improved PSO. *Math. Probl. Eng.* **2015**, *2015*, Article ID 727218, doi:10.1155/2015/727218.
19. Ibrahim, A.; Tawhid, M. A Hybridization of Cuckoo Search and Particle Swarm Optimization for Solving Nonlinear Systems. *Evol. Intell.* **2019**, *12*, 541–561, doi:10.1007/s12065-019-00255-0.
20. Ibrahim, A.; Tawhid, M. A hybridization of Differential Evolution and Monarch Butterfly Optimization for Solving Systems of Nonlinear Equations. *J. Comput. Des. Eng.* **2019**, *6*, 354–367, doi:10.1016/j.jcde.2018.10.006.
21. Liao, Z.; Gong, W.; Wang, L.; Yan, X.; Hu, C. A Decomposition-based Differential Evolution with Reinitialization for Nonlinear Equations Systems. *Knowl. Based Syst.* **2020**, *191*, 105312, doi:10.1016/j.knosys.2019.105312.
22. Kamsyakawuni, A.; Sari, M.; Riski, A.; Santoso, K. Metaheuristic Algorithm Approach to Solve Non-linear Equations System with Complex Roots. *J. Phys. Conf. Ser.* **2020**, *1494*, 012001, doi:10.1088/1742-6596/1494/1/012001.
23. Broer, H.; Takens, F. *Dynamical Systems and Chaos*; Springer-Verlag: New York, NY, USA, 2011.
24. Boeing, G. Visual Analysis of Nonlinear Dynamical Systems: Chaos, Fractals, Self-Similarity and the Limits of Prediction. *Systems* **2016**, *4*, 37, doi:10.3390/systems4040037.
25. Kalantari, B. *Polynomial Root-Finding and Polynomiography*; World Scientific: Singapore, 2009, doi:10.1142/9789812811837.
26. Gościniak, I.; Gdawiec, K. Control of Dynamics of the Modified Newton-Raphson Algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2019**, *67*, 76–99, doi:10.1016/j.cnsns.2018.07.010.
27. Ardelean, G. A Comparison Between Iterative Methods by Using the Basins of Attraction. *Appl. Math. Comput.* **2011**, *218*, 88–95, doi:10.1016/j.amc.2011.05.055.
28. Petković, I.; Rančić, L. Computational Geometry as a Tool for Studying Root-finding Methods. *Filomat* **2019**, *33*, 1019–1027, doi:10.2298/FIL1904019P.
29. Chun, C.; Neta, B. Comparison of Several Families of Optimal Eighth Order Methods. *Appl. Math. Comput.* **2016**, *274*, 762–773, doi:10.1016/j.amc.2015.10.092.

30. Chun, C.; Neta, B. Comparative Study of Methods of Various Orders for Finding Repeated Roots of Nonlinear Equations. *J. Comput. Appl. Math.* **2018**, *340*, 11–42, doi:10.1016/j.cam.2018.02.009.
31. Kalantari, B. Polynomiography: From the Fundamental Theorem of Algebra to Art. *Leonardo* **2005**, *38*, 233–238, doi:10.1162/0024094054029010.
32. Gościniak, I.; Gdawiec, K. One More Look on Visualization of Operation of a Root-finding Algorithm. *Soft Comput.* in press. doi:10.1007/s00500-020-04784-0.
33. Nammanee, K.; Noor, M.; Suantai, S. Convergence Criteria of Modified Noor Iterations with Errors for Asymptotically Nonexpansive Mappings. *J. Math. Anal. Appl.* **2006**, *314*, 320–334, doi:10.1016/j.jmaa.2005.03.094.
34. Gilbert, W. Generalizations of Newton's Method. *Fractals* **2001**, *9*, 251–262, doi:10.1142/S0218348X01000737.
35. Cordero, A.; Torregrosa, J. Variants of Newton's Method using Fifth-order Quadrature Formulas. *Appl. Math. Comput.* **2007**, *190*, 686–698, doi:10.1016/j.amc.2007.01.062.
36. Magreñán, A.; Argyros, I. *A Contemporary Study of Iterative Methods: Convergence, Dynamics and Applications*; Academic Press: San Diego, CA, USA, 2018.
37. Picard, E. Mémoire sur la théorie des équations aux dérivées partielles et la méthode des approximations successives. *J. De Mathématiques Pures Et Appliquées* **1890**, *6*, 145–210.
38. Mann, W. Mean Value Methods in Iteration. *Proc. Am. Math. Soc.* **1953**, *4*, 506–510, doi:10.1090/S0002-9939-1953-0054846-3.
39. Ishikawa, S. Fixed Points by a New Iteration Method. *Proc. Am. Math. Soc.* **1974**, *44*, 147–150, doi:10.1090/S0002-9939-1974-0336469-5.
40. Agarwal, R.; O'Regan, D.; Sahu, D. Iterative Construction of Fixed Points of Nearly Asymptotically Nonexpansive Mappings. *J. Nonlinear Convex Anal.* **2007**, *8*, 61–79.
41. Gdawiec, K.; Kotarski, W. Polynomiography for the Polynomial Infinity Norm via Kalantari's Formula and Nonstandard Iterations. *Appl. Math. Comput.* **2017**, *307*, 17–30, doi:10.1016/j.amc.2017.02.038.
42. Das, G.; Debata, J. Fixed Points of Quasinonexpansive Mappings. *Indian J. Pure Appl. Math.* **1986**, *17*, 1263–1269.
43. Khan, S.; Cho, Y.; Abbas, M. Convergence to Common Fixed Points by a Modified Iteration Process. *J. Appl. Math. Comput.* **2011**, *35*, 607–616, doi:10.1007/s12190-010-0381-z.
44. Gdawiec, K. Fractal Patterns from the Dynamics of Combined Polynomial Root Finding Methods. *Nonlinear Dyn.* **2017**, *90*, 2457–2479, doi:10.1007/s11071-017-3813-6.
45. Su, Y.; Qin, X. Strong Convergence of Modified Noor Iterations. *Int. J. Math. Math. Sci.* **2006**, *2006*, 21073, doi:10.1155/IJMMS/2006/21073.